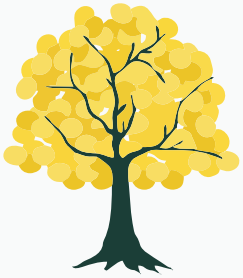
02 a 05
de outubro 2017
Uberlândia - MG

SBBD

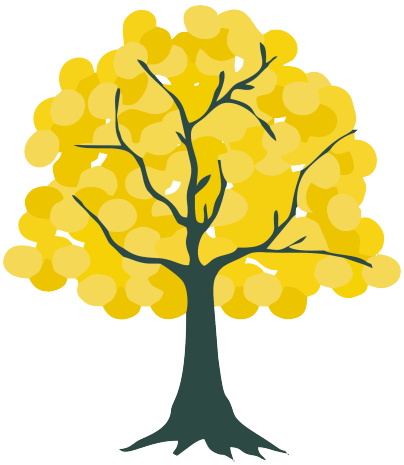
TÓPICOS EM GERENCIAMENTO DE DADOS E INFORMAÇÕES 2017

Vaninha Vieira, Humberto L. Razente,
Maria Camila N. Barioni (Org.)

Sociedade Brasileira de Computação



SBBD2017



32nd BRAZILIAN SYMPOSIUM ON DATABASES
October 2nd to 5th, 2017
Uberlândia – MG – Brazil

TÓPICOS EM GERENCIAMENTO DE DADOS E INFORMAÇÕES 2017

Editora
Sociedade Brasileira de Computação – SBC

Organizadores
Vaninha Vieira
Humberto Luiz Razente
Maria Camila Nardini Barioni

Realização
Sociedade Brasileira de Computação – SBC
Comissão Especial de Banco de Dados (CEBD) da SBC
Universidade Federal de Uberlândia – UFU

ISBN: 978-85-7669-400-7

Dados Internacionais de Catalogação na Publicação (CIP)

B827t Brazilian Symposium on databases (32. : 2017 : Uberlândia, MG, Brazil)

Tópicos em gerenciamento de dados e informações 2017 [recurso eletrônico], 2 a 5 de Outubro de 2017 em Uberlândia, Minas Gerais ; organizadores Vaninha Vieira, Humberto Luiz Razente, Maria Camila Nardini Barioni. - Uberlândia: SBC, 2017.

94 p. : il.

ISBN: 9788576694007

Inclui bibliografia.

Modo de acesso: http://www.sbbd.org.br/2017

1. Banco de dados - Congressos. 2. Bases de Dados - Congressos. I. Vieira, Vaninha. II. Razente, Humberto Luiz. III. Barioni, Maria Camila Nardini. IV. Universidade Federal de Uberlândia. V. Sociedade Brasileira de Computação. VI. Título.

CDU: 681.3(061.3)

Editorial

Os capítulos deste livro foram escritos pelos autores dos minicursos apresentados no XXXII Simpósio Brasileiro de Banco de Dados (SBBD 2017) e no V *Symposium on Knowledge Discovery, Mining and Learning* (KDMiLe 2017). Os minicursos têm como objetivo apresentar temas relevantes da área de Banco de Dados e promover discussões sobre os fundamentos, tendências e desafios relacionados ao tema abordado. Os minicursos têm três horas e meia de duração e constituem uma excelente oportunidade de atualização para acadêmicos e profissionais que participam do evento.

Três minicursos foram selecionados e compõem este livro. Os dois primeiros foram escolhidos pela comissão de minicursos do SBBD 2017, em processo de avaliação que contou com, ao menos, 3 revisores por proposta submetida, observando a não existência de conflitos de interesse entre o revisor e a proposta. O tema do terceiro minicurso, associado ao KDMiLE 2017, foi proposto pelo comitê diretivo, e o professor autor do minicurso foi convidado pela coordenação local do evento com a concordância do comitê diretivo.

O Capítulo 1 apresenta o minicurso "É uma questão de tempo! Extraindo Conhecimento de Redes Sociais Temporais". Esse minicurso discute técnicas de visualização e processamento de dados extraídos de redes sociais, estruturados em forma de rede, considerando especialmente a perspectiva temporal, e apresenta a prática do uso dessas técnicas por meio da análise de três estudos de caso com dados reais.

No Capítulo 2 temos o minicurso "*Sports Analytics*: Mudando o Jogo", que discute técnicas de modelagem de predição e descoberta de conhecimento, e sua influência em um domínio específico: dados relacionados a jogos esportivos. O minicurso discute, ainda, o papel da preditibilidade e aleatoriedade nos esportes.

O Capítulo 3 traz o minicurso "Como funciona o *Deep Learning*". Esse minicurso descreve os conceitos básicos de *Deep Learning*, ilustra, por meio de exemplos, como implementar redes profundas e os principais desafios em treinar esse tipo de rede, e apresenta as bases teóricas por trás do uso de modelos profundos, e suas limitações.

Gostaríamos de agradecer aos autores pela submissão das propostas e geração dos textos finais, e ao Comitê de Avaliação, pela dedicação e eficiência no processo de seleção.

Vaninha Vieira (UFBA)
Coordenadora de Minicursos do SBBD 2017

Humberto Luiz Razente (UFU)
Maria Camila Nardini Barioni (UFU)
Coordenadores Locais do SBBD 2017

XXXII Simpósio Brasileiro de Banco de Dados

02 a 05 de Outubro 2017
Uberlândia – MG – Brasil

MINICURSOS

Promoção

Sociedade Brasileira de Computação – SBC
Comissão Especial de Banco de Dados (CEBD) da SBC

Organização

Faculdade de Computação, Universidade Federal de Uberlândia – UFU

Comitê Diretivo do SBBD 2017

Javam Machado (UFC), Coordenador da CEBD
Agma Juci Machado Traina (USP)
Bernadette Lóscio (UFPE)
Caetano Traina Jr. (USP)
Carmem Hara (UFPR)
Mirella M. Moro (UFMG)
Vanessa Braganholo (UFF)

Coordenadores do SBBD 2017

Coordenador do Comitê Diretivo
Javam Machado (UFC)

Coordenadores de Organização Local
Maria Camila Nardini Barioni (UFU) e Humberto Luiz Razente (UFU)

Coordenadora do Comitê de Programa
Carmem S. Hara (UFPR)

Coordenadoras do Comitê de Programa de Artigos Curtos
Bernadette Lóscio (UFPE) e Damires Souza (IFPB)

Coordenador da Sessão de Demos e Aplicações
Daniel de Oliveira (UFF)

Coordenadora do Workshop de Teses e Dissertações em Banco de Dados
Carina Dorneles (UFSC)

Coordenadora de Minicursos
Vaninha Vieira (UFBA)

Coordenadora de Tutoriais
Ana Carolina Salgado (UFPE)

Coordenadora do Concurso de Teses e Dissertações
Vânia Vidal (UFC)

Coordenadora de Workshops
Fernanda Baião (UNIRIO)

Comitê Diretivo do KDMiLe 2017

Alexandre Plastino (UFF) (Coordenador)
André Ponce de Leon F. de Carvalho (ICMC-USP)
Wagner Meira Jr. (UFMG)

Comitê de Organização Local

Coordenadora Local do KDMiLe 2017
Elaine Ribeiro de Faria (UFU)

Membros do Comitê de Organização Local
Maria Camila N. Barioni (UFU)
Humberto L. Razente (UFU)

Elaine Ribeiro de Faria Paiva (UFU)
João Henrique de Souza Pereira (UFU)
José Gustavo de Souza Paiva (UFU)
Marcelo Zanchetta do Nascimento (UFU)

Comitê de Avaliação de Minicursos

Vaninha Vieira (UFBA) (Coordenadora)
Daniela Barreiro Claro (UFBA)
João Batista Rocha Jr. (UEFS)
José Palazzo Moreira de Oliveira (UFRGS)

Sumário

mini:1

Capítulo 1

É uma questão de tempo! Extraindo Conhecimento de Redes Sociais Temporais

Fabíola S. F. Pereira, João Gama, Gina M. B. de Oliveira

Abstract

Data is structured as a network. And now? How to analyze it? Extracting knowledge from network data is not a simple task and requires the use of appropriate tools and techniques, especially in scenarios that take into account the volume and evolving aspects of the network. In this chapter it is considered that data has already been collected and is already structured as a network. The goal is to discuss techniques to analyze this network data, especially considering the time perspective. First, concepts related to problem definition, temporal networks and metrics for network analysis will be presented. Next, in a more practical aspect will be shown techniques of visualization and processing of temporal networks. In the end, three case studies with real data will be discussed, illustrating how network data knowledge extraction works from start to finish.

Resumo

Os dados estão estruturados na forma de rede. E agora? Como analisá-los? Extrair conhecimento desse tipo de dado não é uma tarefa simples e requer o uso de ferramentas e técnicas adequadas, especialmente em cenários que levam em conta o volume de dados e o aspecto temporal da rede. Neste capítulo considera-se que os dados já foram coletados e já estão estruturados em forma de rede e discute-se sobre técnicas para analisá-los, considerando especialmente a perspectiva temporal. Primeiro serão apresentados conceitos relacionados à definição do problema, redes temporais e métricas para análise de rede. Em seguida, em um aspecto mais prático serão mostradas técnicas de visualização e processamento de redes temporais. Ao final, três estudos de caso com dados reais serão discutidos, ilustrando do começo ao fim como funciona a extração de conhecimento de dados em rede.

1.1. Introdução

Redes sociais de amizade, redes de *hyperlinks*, de confiança e redes de co-autoria são exemplos de dados estruturados na forma de redes que representam entidades ligadas por alguma relação em comum. Análise de redes sociais é o campo de estudo que busca entender a estrutura e comportamento dessas redes, bem como as entidades que a ela pertencem [24]. Recentemente, houve um crescente interesse da comunidade de mineração de dados nesse campo de análise de redes sociais. A motivação básica é a demanda por explorar o conhecimento de grandes volumes de dados coletados, pertencentes ao comportamento social dos usuários em ambientes *online* [30].

Existe uma vasta literatura acerca de como coletar, pré-processar e modelar dados de mídias sociais em forma de redes [8], bem como acerca das principais métricas de centralidade [1]. Porém, ainda há muito a ser discutido em relação à análise da rede obtida. Por onde começar a análise de uma rede? E se ela for muito grande, como visualizá-la? Como apresentar os resultados obtidos? Quais as melhores técnicas para filtros e processamento? E para considerar a evolução temporal é melhor trabalhar com *snapshots*?

Neste capítulo então, considera-se que os dados já foram coletados e já estão estruturados em forma de rede e discute-se sobre técnicas para analisá-los, considerando especialmente a perspectiva temporal. É uma perspectiva pouco explorada e muito útil dentro do contexto de ciência de dados em rede.

1.1.1. Entendendo e Formalizando o Problema

Ao receber uma coleção de dados estruturados na forma de rede e a informação sobre o domínio que aquela rede representa (ex.: relações de amizade, trocas de e-mails, contatos visuais entre pessoas etc), o primeiro passo que um cientista de dados deve tomar é visualizar a rede recebida. A visualização pode ser feita com o auxílio de ferramentas (Seção 1.4) e, normalmente, sobre uma amostra obtida do todo.

Ao visualizar a rede, é possível rapidamente obter *insights* acerca dos dados em questão, tais como: quais características descrevem um nó e uma aresta, se a rede possui uma configuração parecida com algum modelo antes visto (esparsa, comunidades pequenas, bipartite), se possui informação temporal nas arestas e qual a granularidade dessa informação, se originalmente é dirigida, ou ainda, se originalmente possui diferentes tipos de nós. Note que a visualização de uma rede contempla não só a imagem estrutural, como também, os dados que a formam.

Uma vez observada a rede inicial, o segundo passo de uma análise é voltar ao problema em questão: qual é o problema que deseja-se resolver? Em alguns cenários o problema pode ser claramente solicitado, por exemplo, obter as comunidades da rede; em outros, o problema não está claro, e deve-se começar por uma análise exploratória que, futuramente, ajudará na elaboração de hipóteses para a definição do problema. Uma análise exploratória é, por exemplo, obter algumas medidas de influência, modularizar baseado em diferentes características dos nós, filtrar, observar nós *egos* (ego network [30]) etc. Dependendo do cenário, um análise exploratória já é suficiente para quem forneceu a rede. A entrega é, portanto, um conjunto de estatísticas que descrevem a rede em questão.

Um passo adiante da análise de exploratória é, finalmente, levar em consideração o aspecto temporal. Mesmo que o problema tenha claramente indicado que o tempo é fator fundamental, é importante passar pelas etapas básicas de conhecimento dos dados primeiro. Elas ajudam a perceber a real necessidade do tempo ser considerado. Em geral, problemas que consideram redes temporais estão relacionados à extração de conhecimento que leva em conta o fluxo natural de evolução da rede, ou o fluxo de informações que propagam na rede [28, 27, 23]. Por exemplo, é natural entender uma rede de contatos (aperto de mãos) entre pessoas em um contexto de transmissão de doenças. A Figura 1.1 sumariza as ideias até então apresentadas.

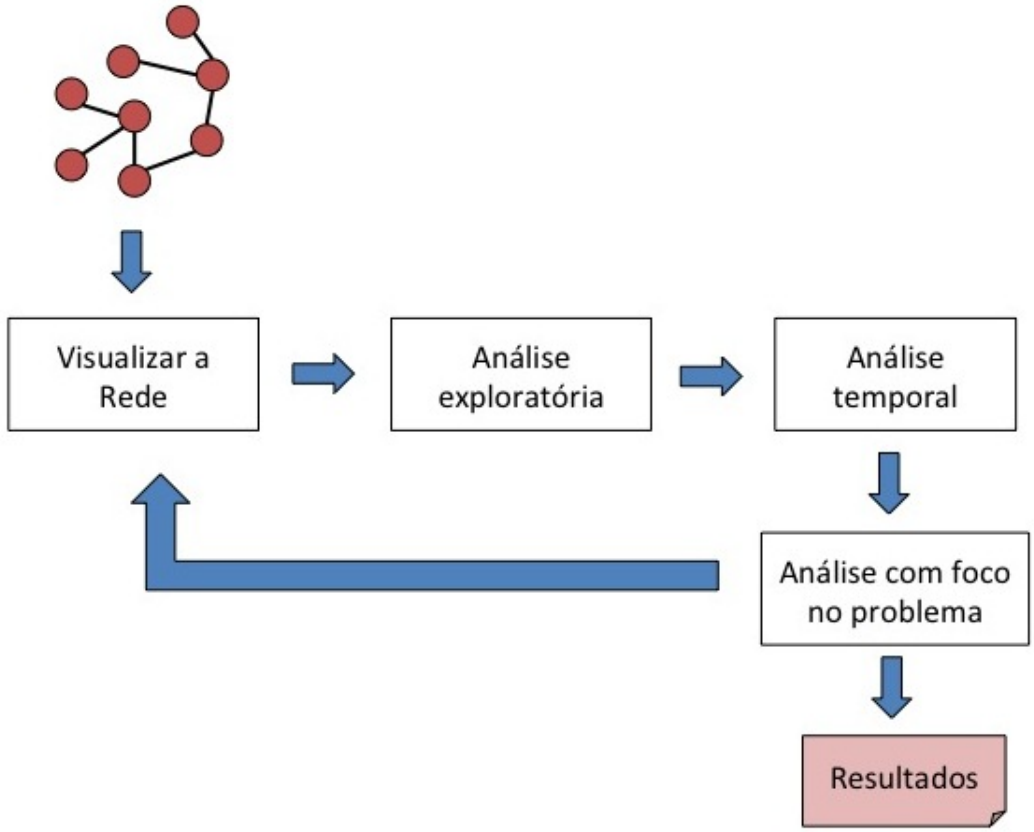


Figura 1.1. Passos para entender o problema da descoberta de conhecimento em redes.

É importante seguir tais passos para manter a visão crítica e conseguir discernir se uma rede que envolve a complexidade de uma análise temporal é realmente necessária. Ao longo do capítulo considera-se que o fluxo de descoberta de conhecimento em redes é respeitado e executado de maneira recorrente, até que se consiga respostas para um determinado problema.

1.1.2. Organização do capítulo

Na Seção 1.2 são apresentados os conceitos acerca de redes sociais temporais, bem como as principais métricas para analisá-las. Na Seção 1.3 são apresentadas as diferentes estratégias para processar redes evolutivas durante a análise. Tais estratégias variam entre processamento em blocos ou processamento de *streams* de redes. A Seção 1.4 é uma sín-

tese sobre estratégias de visualização de redes sociais temporais. A Seção 1.5 descreve três ferramentas principais para análise de redes. Na Seção 1.6 são apresentados três estudos de caso com bases de dados reais, destacando os processos utilizados na análise de cada rede para extração de conhecimento. Por fim, a Seção 1.7 apresenta as considerações finais.

1.2. Redes Sociais Temporais

Estruturas de redes representam relacionamentos entre entidades. Nas redes temporais, os tempos em que esses relacionamentos estão ativos são elementos explícitos da representação [11]. Um exemplo clássico de aplicação de uma rede temporal é o contágio de doenças através da proximidade física. Comumente, a propagação de organismos patogênicos ocorre através de um aperto de mão e uma rede temporal é a melhor estrutura para representar esse cenário. Redes sociais, tópico de interesse deste capítulo, também podem ser representadas como redes temporais, já que estão cada vez mais ubíquas e complexas em suas interações [10].

Existem várias definições na literatura que formalizam redes temporais (aqui, invariavelmente chamadas também de grafos temporais) [29, 17, 15, 11]. [15] definiu os grafos ordenados no tempo e [17] os chama de grafos que variam com o tempo, mas geralmente todas as definições representam um conjunto de arestas temporais e um conjunto de nós durante um intervalo de observação que leva em conta a ordem temporal em que aparecem (ou estão ativos).

Definição 1 (Rede temporal) *Uma rede temporal $G = (V, E)$ é um conjunto E de arestas registradas em meio a conjunto de nós V durante um intervalo de observação $[0, T]$. Uma aresta entre dois nós $u, v \in V$ é representada por uma quádrupla $e = (u, v, t, \delta t)$, onde $0 \leq t \leq T$ é o momento que a aresta surgiu e δt é sua duração. As arestas também são chamadas de contatos ou ligações.*

Essa definição é clássica para representar grafos de voos e redes de chamadas telefônicas, por exemplo. Mas existem extensões para a definição acima. Quando os contatos são instantâneos, $\delta t \to 0$, a rede temporal é definida como um grafo de sequência de contatos [11]. Esses grafos são usados para representar sistemas cuja duração do contato é menos importante (redes de e-mails, sexuais, redes de *likes* em redes sociais). Outra variação ao invés de definir redes temporais com arestas que não estão ativas sobre um conjunto de instantes, é defini-las sobre um conjunto de intervalos $e = (u, v, t_{init}, t_{end})$. Estes são os conhecidos grafos de intervalo, bons para modelagem de relacionamentos do tipo seguidor/seguido no Twitter [22]. De fato, grafos de intervalos podem ser transformados em grafos de sequência de contatos e, então, a maioria das técnicas de análise de redes pode ser empregada independente da modelagem utilizada.

Exemplo 1 *A Figura 1.2 ilustra duas redes temporais, considerando o contexto da rede social Twitter. A Figura 1.2(a) é um grafo de sequência de contatos, representando menções entre os usuários. Os nós são usuários e uma aresta (u, v, t) indica que u mencionou v em um tweet postado no tempo t*[1]*. Os instantes que ocorrem as interações estão descri-*

[1]Menção é um tweet que contém uma referência a outro usuário

tos próximos às arestas e a duração das interações é negligenciável. É possível perceber que os usuários A e B interagiram nos instantes 3, 6 e 11, os usuário B e C interagiram em 7 e 9 e assim por diante.

Agora, no mesmo contexto do Twitter, é possível considerar um grafo de intervalos na Figura 1.2(b), onde as arestas representam relações de seguidores/seguidos e os intervalos indicam que tais relações começaram em t_{init} e terminaram em t_{end}. Como exemplo, E começou a seguir F em 3 e deixou de segui-lo em 6.

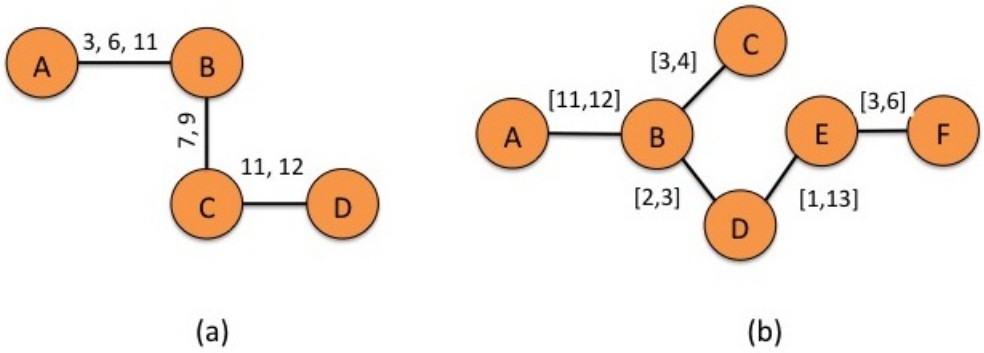


Figura 1.2. Redes temporais representadas como (a) sequência de contatos e (b) grafo de intervalos.

1.2.1. Métricas Temporais de Redes

A estrutura topológica de redes estáticas pode ser caracterizada por uma série de métricas [7, 26]. Em essência, tais medidas são baseadas em conexões entre nós vizinhos (tais como *degree* ou coeficiente de clusterização), ou entre grandes conjuntos de nós (caminhos, diâmetro e métricas de centralidade). Quando a dimensão tempo é incluída na rede, muitas dessas medidas precisam ser repensadas.

1.2.1.1. Caminhos em função do tempo

Em um grafo estático, um caminho é simplesmente uma sequência de arestas tais que uma aresta termina no nó onde a próxima aresta inicia (tal como o caminho A para B para C para D na Figura 1.2. Em um grafo temporal, caminhos são definidos como sequências de contatos com tempos crescentes que conectam um conjunto de nós – os caminhos em função do tempo [14]. Como exemplo, na Figura 1.2(b) existe um caminho em função do tempo de A para B (<(A,B,11)>, por exemplo), mas nenhum de A para D.

Uma diferença entre redes estáticas e temporais é que os caminhos não são transitivos. A existência de um caminho em função do tempo de i para j e de j para k não implica na existência de um caminho de i para k. Esse fato está relacionado a uma propriedade fundamental nos caminhos em função do tempo – um caminho de i para k via j existe somente se o primeiro contato entre j e k ocorreu depois de um contato em i e j.

Portanto, caminhos em função do tempo definem quais nós podem ser atingidos a partir de outros nós dentro de uma janela de observação $t \in [0,T]$. O conjunto de nós

que podem ser atingidos a partir de um nó i é chamado de conjunto de influência de i. No contexto de redes sociais, por exemplo, o conjunto de influência será atingido pelos *posts* de i.

A duração de um caminho em função do tempo é a diferença entre o último e o primeiro contato de um caminho [20]. Analogamente ao conceito de menores caminhos em redes estáticas que define a distância geodésica, em redes temporais existem os *caminhos mais rápidos em função do tempo*, que indicam caminhos com menor duração. No exemplo (Figura 1.2)(b) existem vários caminhos em função do tempo de B para F na janela de observação $[3,15]$: <(B,D,2),(D,E,4),(E,F,5)>, <(B,D,3),(D,E,4),(E,F,6)>, etc.. O caminho mais rápido em função do tempo possui duração 3.

1.2.1.2. Medidas de Centralidade Temporal

Na teoria de rede, diversas medidas de centralidade foram definidas para identificar o comportamento dos nós e arestas, muitas delas fundamentadas no conceito de menores caminhos. Trazendo para o cenário de redes temporais, a ideia é interpretar tais medidas utilizando os caminhos mais rápidos em função do tempo (ao invés de menores caminhos).

Closeness temporal. A métrica de centralidade closeness C_C [5] para redes estáticas é definida como

$$C_C(i) = \frac{N-1}{\sum_{j \neq i} d(i,j)} \tag{1}$$

onde $d(i,j)$ é a menor distância geodésica entre i e j, i.e. a centralidade closeness de um nó mede o inverso da menor distância total para todos os outros nós e é alta para aqueles nós mais próximos de todos os outros (centrais). Similarmente, em redes temporais, a ideia é medir o quão rápido um nó pode em média atingir os demais:

$$C_C(i,t) = \frac{N-1}{\sum_{j \neq i} \lambda_{i,t}(j)} \tag{2}$$

onde $\lambda_{i,t}(j)$ é a latência entre i e j, definida por $\lambda_{i,t}(j) = t - \phi_{i,t}(j)$, sendo $\phi_{i,t}(j)$ o instante mais recente antes de t em que a informação de j pode ter atingido i. A latência mede a idade da informação em um nó.

Betweenness temporal. A centralidade betweenness C_B [5] é também baseada em menores caminhos. Ela mede a fração entre o número de menores caminhos que passam pelo nó em questão em função do número total de menores caminhos entre cada par de nós na rede. Para redes estáticas, tal centralidade é formalmente definida por

$$C_B(i) = \frac{\sum_{i \neq j \neq k} v_i(j,k)}{\sum_{i \neq j \neq k} v(j,k)} \tag{3}$$

onde $v_i(j,k)$ é o número de menores caminhos entre j e k que passam por i, e $v(j,k)$ é o número total de menores caminhos entre j e k. Pensando no contexto de redes temporais, tem-se:

$$C_B(i,t) = \frac{\sum\limits_{i \neq j \neq k} w_{i,t}(j,k)}{\sum\limits_{i \neq j \neq k} w_t(j,k)} \tag{4}$$

onde $w_{i,t}(j,k)$ é o número de caminhos mais rápidos em função do tempo na janela de observação t entre j e k que passam por i e $w_t(j,k)$ é a quantidade total de caminhos mais rápidos em função do tempo.

Além das métricas mais populares de closeness e betweenness, em [11] são discutidas diversas outras métricas que se aplicam ao contexto temporal.

1.3. Análise de Redes Evolutivas

Quando o assunto são redes que evoluem ao longo do tempo, uma questão importante é discutir como processá-las. Por exemplo, em redes de e-mails as arestas são adicionadas a cada minuto, enquanto em redes de co-autoria as arestas surgem em escalas de semanas ou meses. Assim, mesmo considerando os aspectos de evolução e informação temporal, é necessário refletir se análises *offline* ou em tempo real são necessárias em determinado contexto.

1.3.1. Estratégias para Manipular a Evolução da Rede

Aqui estão sumarizadas diferentes estratégias de processamento de redes que evoluem ao longo do tempo. Essas estratégias estão relacionadas com a rede – ou amostras dela – que são consideradas durante a análise, impactando (i) no processo de ajuste do modelo, (ii) performance dos algoritmos e (iii) interpretação semântica que o analista está interessado. O termo *ordem temporal das arestas* se refere à ordem em que as arestas chegam na rede, ou seja, se uma rede temporal é considerada na análise, conforme discutido nas seções anteriores. Na Figura 1.3 está ilustrado um cenário em que as arestas chegam em fluxo (*edge stream*) que será usado para exemplificar as estratégias.

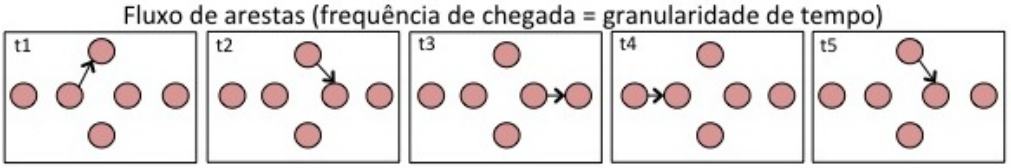


Figura 1.3. Fluxo de arestas

- *Redes que evoluem lentamente (slowly evolving networks).* A maioria dos trabalhos propostos na década passada considera essas estratégias quando processam redes [9]. Elas são intuitivas e diretas, conforme mostrado na Figura 1.4.

– *Processamento em blocos (batch processing).* Toda a rede é simplesmente processada considerando a ordem temporal das arestas. Algoritmos clássicos como Dijkstra são usados nesse cenário.

– *Snapshots.* A cada instante $t1, t2, ...$ um snapshot da rede é considerado. Aqui, a ordem temporal só faz sentido se a granularidade dos snapshots for maior que a granularidade da ordem de chegada das arestas.

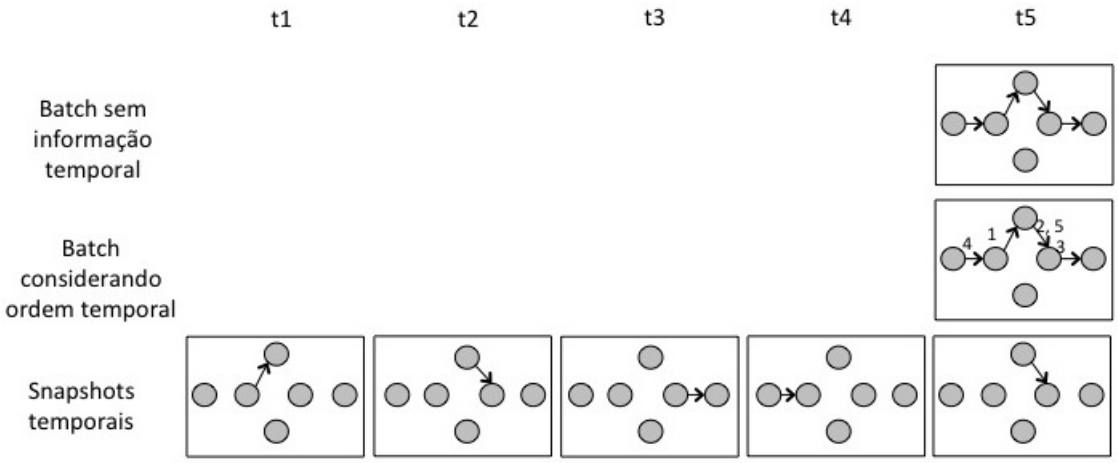


Figura 1.4. Estratégias para processamente de redes que evoluem lentamente considerando o fluxo de arestas apresentado na Figura 1.3

- *Redes stream.* Esse cenário é muito mais desafiador em termos de algoritmos devido a restrições computacionais e à incapacidade de carregar toda a rede no disco. O que varia é a abordagem de janelas e se a ordem temporal das arestas também é levada em conta dentro das janelas (Figura 1.5)). Independente da estratégia, em redes stream os algoritmos deve usar estruturas de dados que podem ser mantidas incrementalmente.

  – *Janela landmark.* Essa estratégia é boa quando deseja-se manter o histórico e as novas arestas que chegam são processadas considerando todo o grafo armazenado até então. Levar em conta a ordem temporal (redes temporais) é um cenário muito comum. Por exemplo, redes de contato entre pessoas são processadas utilizando essa estratégia num contexto de propagação de doenças [11].

  – *Janela deslizante (sliding window).* Na janela deslizante o passado recente da rede é suficiente. Essa é a estratégia base para algoritmos de amostragem com fator de esquecimento [2]. Não é usual considerar a ordem temporal dentro da janela.

  – *Janela de observação fixa.* Alguns trabalho processam a rede com uma janela de observação fixa [29], na qual apenas um determinado intervalo é interessante. Grafos de voos são processados usando essa estratégia, considerando redes temporais dentro da janela.

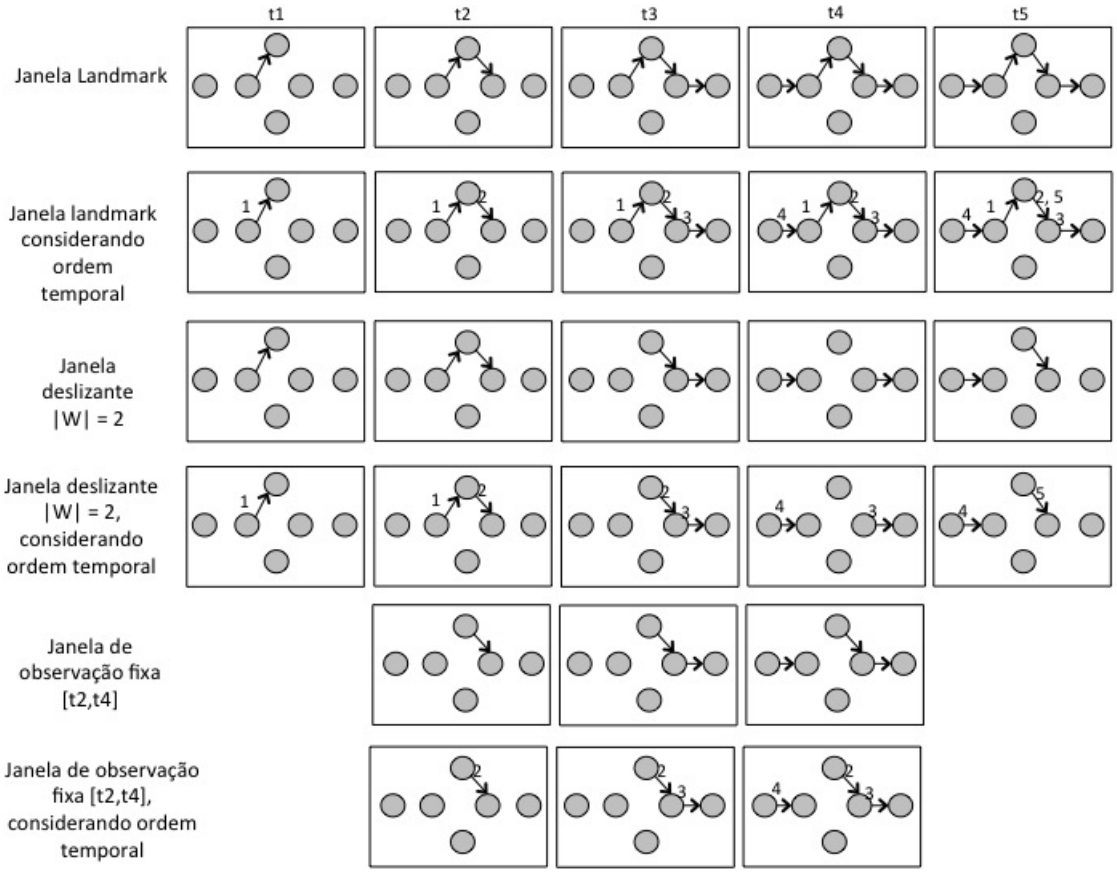


Figura 1.5. Estratégias para processamento de redes stream considerando o fluxo de arestas da Figura 1.3

1.3.2. Nivelando os conceitos

Não existe um consenso na literatura em relação aos termos utilizados para expressar redes que evoluem ao longo do tempo. Os conceitos mais frequentemente empregados são:

- Rede evolutiva (*evolving network, evolutionary network*) ou rede evolutiva no tempo (*time-evolving network*) ou rede dinâmica (*dynamic network*) ou rede com variação no tempo (*time-varying network*). Todos esses conceitos referem-se a redes que estão mudando, com nós aparecendo e desaparecendo, associando e desassociando uns com os outros à medida que o tempo passa.
- Rede temporal. Como detalhado na Seção 1.2, redes temporais são redes cuja ordem em que as arestas aparecem e desaparecem é levada em conta durante a análise – a ordem temporal.
- Rede *stream*. Esse termo está relacionado à maneira que uma rede é observada e processada, especialmente em cenários onde não existe a noção de começo e fim do fluxo de chegada dos dados.

1.4. Visualização de Redes

O uso de recursos visuais pode auxiliar na obtenção de *insights* durante o processo de análise. De fato, técnicas de visualização de redes têm sido muito exploradas atualmente [3, 16, 12, 18].

A visualização auxilia principalmente no processo inicial da análise de uma rede temporal. Por exemplo, considere que diante de uma rede na qual apenas o domínio é conhecido – rede temporal de contatos entre pessoas e possível transmissão de doenças. A Figura 1.6 é um exemplo de visualização que pode ser aplicada na rede. A partir da figura é possível obter os seguintes *insights*:

1. A pessoa D inicialmente é muito ativa, mas à medida que o tempo passa ela deixa de entrar em contato com as demais pessoas;
2. As pessoas B e C são muito ativas na rede;
3. A pessoa A tem baixo nível de contatos;
4. Se a pessoa A possuir o vírus da gripe na altura do tempo $t = 6$, apesar do seu baixo nível de interatividade, D nunca será infectado, porém B e C serão.

Se a visualização temporal, entretanto, não tivesse sido utilizada, provavelmente a rede de contatos entre pessoas seria representada como na Figura 1.7. Apenas olhando para a rede estática, não é possível extrair informações da sequência de contatos e, erroneamente, poderia-se concluir que se A possuir vírus da gripe, todos serão infectados.

Primordialmente, a visualização é, então, utilizada para obtenção de insights por parte do analista da rede. Porém, as ferramentas de visualização são acionadas também no

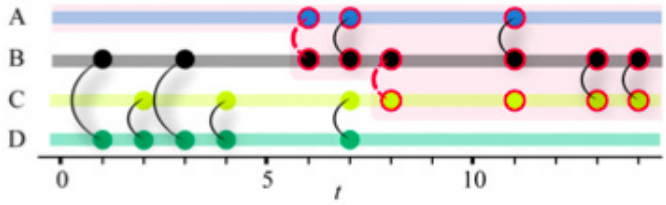


Figura 1.6. Obtida de [11]. Representação explícita da dimensão temporal da rede que ilustra uma sequência de contatos entre pessoas e possível transmissão de doença na rede.

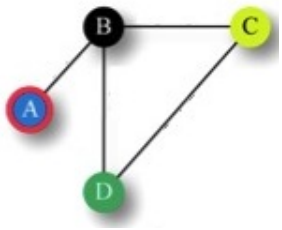


Figura 1.7. Adaptada de [11]. Rede estática de contatos entre pessoas.

momento da apresentação de resultados. Considere, por exemplo, que ao final da análise de uma rede temporal, queira-se destacar o padrão de evolução da centralidade de um nó. As técnicas de visualização buscam maneiras de representar tal nó (ou comunidades) em destaque no meio de uma infinidade de nós e arestas se cruzando.

1.5. Ferramentas para Análise

As principais ferramentas para análise e visualização de redes sociais temporais são enumeradas a seguir:

1. Gephi [4]. Ferramenta *open source* construída em Java, com arquitetura flexível, pronta para receber plugins. Essa flexibilidade tem tornado o Gephi a ferramenta mais popular para análise de redes dinâmicas. A Figura 1.8 ilustra uma rede social desenhada através dessa ferramenta e a Figura 1.9 destaca a funcionalidade temporal disponível no Gephi.

2. ORA [6]. A principal característica dessa ferramenta é a organização e disponibilização de funcionalidades, que estão dispostas com um nível maior de abstração. Por exemplo, ao invés de ter funcionalidades como calcula *closeness*, *betweenness*, ORA possui funcionalidades como *obtém mais influentes*, *obtém nós que são pontes* e assim por diante. É uma ferramenta cujo público-alvo não precisa ser técnico para conseguir manipulá-la, como por exemplo jornalistas. ORA suporta redes sociais padrões (Twitter, Facebook), redes organizacionais, geo-espaciais, *meta-networks*, redes dinâmicas entre outras. A Figura 1.10 é um *snapshot* da ferramenta.

3. DyNetVis [16]. É um software específico para visualização de redes dinâmicas que possui um paradigma diferente das demais ferramentas mencionadas. O DyNetVis utiliza o conceito de layout temporal (ao invés do estrutural) para apresentar a rede.

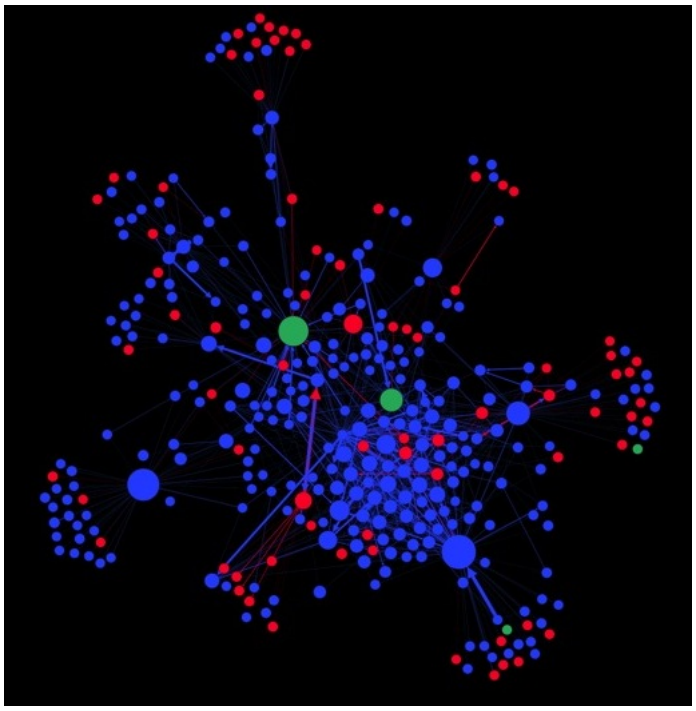


Figura 1.8. Exemplo de rede social obtida através do Gephi.

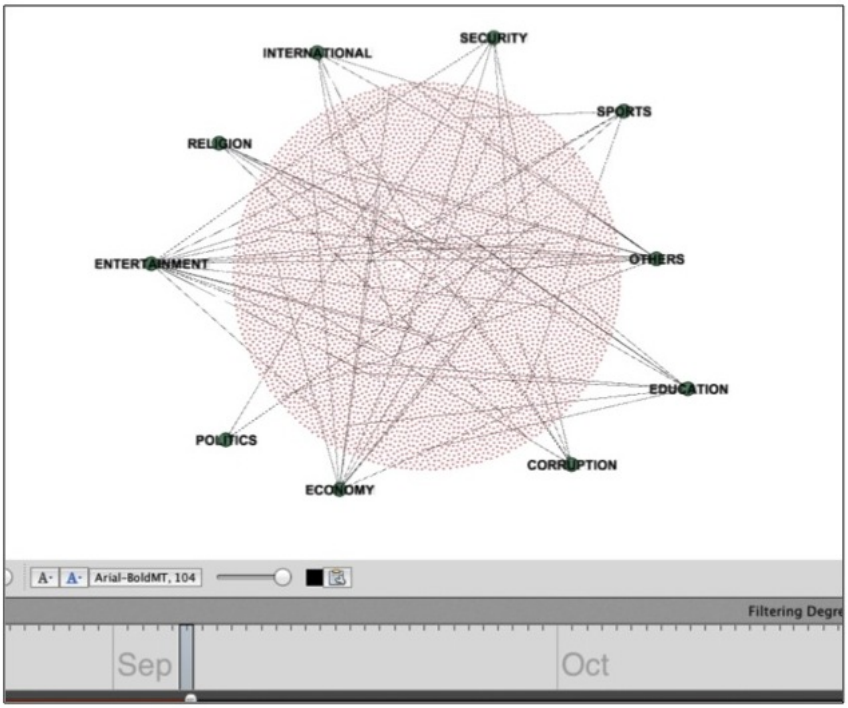


Figura 1.9. Exemplo de rede social temporal sendo analisada no Gephi.

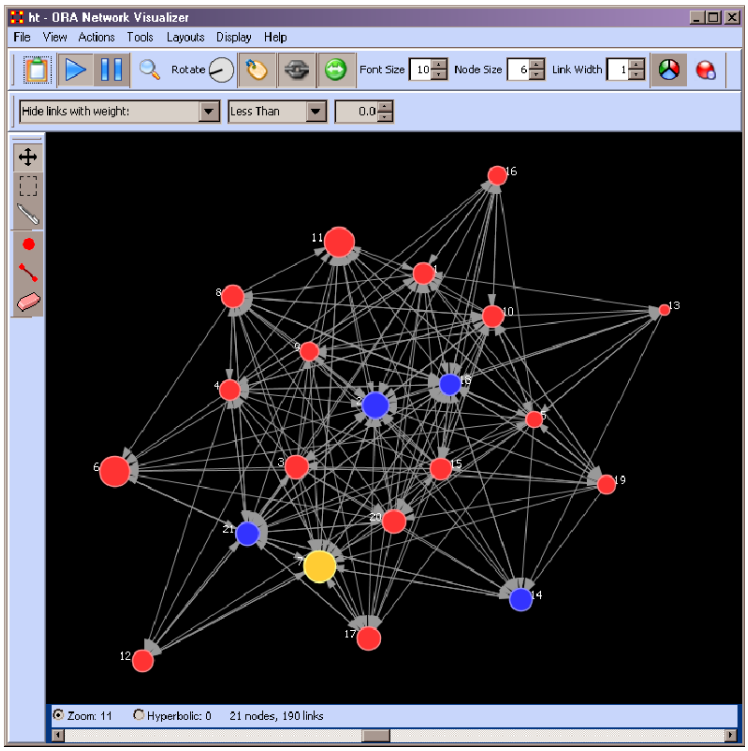


Figura 1.10. Interface da ferramenta ORA [13].

Com isso, é possível por exemplo na etapa da análise exploratória, perceber a evolução da comunicação entre nós vizinhos recorrentes (comunidades). A Figura 1.11 ilustra uma rede temporal visualizada pelo DyNetVis.

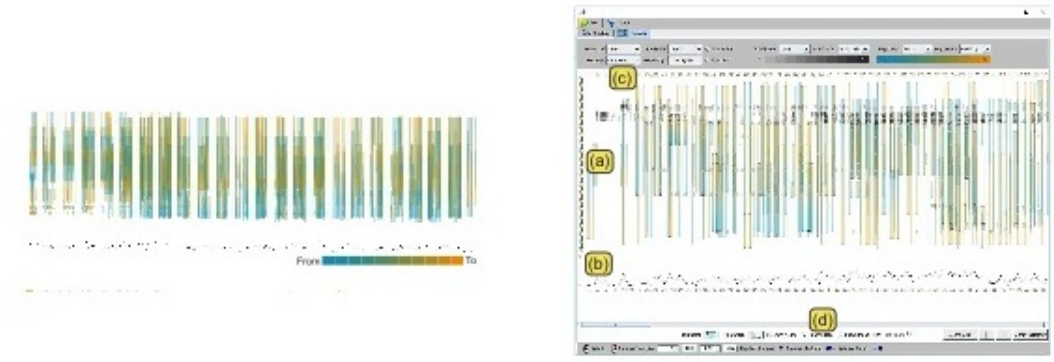


Figura 1.11. Layout temporal produzido pelo DyNetVis [16]

Outras ferramentas comuns para análise de redes sociais são R (pacotes *igraph*, *network*, *sna*), Neo4J e SNAP. Porém, quando o interesse é em análise temporal, essas ferramentas ainda não possuem flexibilidade e funcionalidades superiores àquelas acima descritas.

1.6. Estudos de Caso

Tendo como base os conceitos, estratégias e processos de análise discutidos neste capítulo, são apresentados três estudos de caso com bases de dados reais. O foco do estudo é destacar os processos utilizados na análise de cada rede para obtenção de conhecimento.

1.6.1. Rede Social do Twitter

Considerando o Twitter como domínio, nesse estudo de caso o objetivo é analisar a evolução de interações entre usuários em relação a notícias da Folha de São Paulo. Parte desse estudo foi conduzida no trabalho [21]. Na rede social, os nós são usuários do Twitter e uma aresta dirigida de u para v representa que v retweetou u[2]. A Tabela 1.1 sumariza as estatísticas da rede em questão.

Tabela 1.1. Estatísticas da rede social temporal do Twitter.

Conteúdo da Rede	
Domínio	Notícias da Folha de S. Paulo no Twitter
Política de crawling	tweets com menção @folha
Intervalo de tempo	8/8/2016 - 9/11/2016
# total tweets	1,771,435
# tweets *retweetados*	150,822
Topologia da rede	
# nós	292,310
# arestas temporais (retweets)	1,392,841

O conteúdo de notícias da rede está estruturado através de tópicos que sumarizam o assunto de um determinado tweet. Por exemplo, segurança, corrupção, política, esportes etc. A Figura 1.12 ilustra a evolução de uma amostra da rede. Note que a análise da evolução torna-se mais rica com a inserção das informações sobre os tópicos na rede (mais especificamente, nas arestas).

Considerando a metodologia para entendimento do problema e descoberta de conhecimento em redes (Seção 1.1.1), o próximo passo de análise dessa rede é explorar métricas de centralidade, tanto temporais quanto estáticas. As métricas obtidas estão sumarizadas na Tabela 1.2. Em geral os nós dessa rede possuem baixas centralidades, principalmente closeness, pois os retweets estão sempre vinculados ao usuário que originalmente postou. Não é uma cascata de interações.

Tabela 1.2. Medidas de centralidade da rede social temporal do Twitter.

Métricas da rede	estática	temporal
Média do tamanho do caminho mais curto	12.31	5 dias
Média do grau	4.76	-
Média do closeness	1.01	2.44
Média do betweenness	0.0056	0.0233

[2]Retweet é um compartilhamento que um usuário u faz de um tweet originalmente postado por outro usuário v

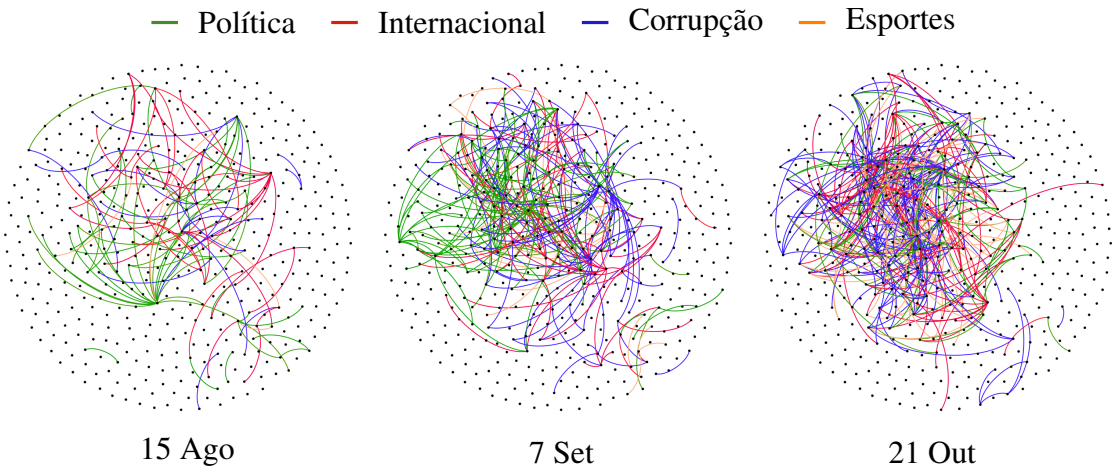


Figura 1.12. *Snapshots* de amostras da rede de retweets. Nós são usuários do Twitter. Uma ligação de u_1 para u_2 significa que u_2 retweetou no instante t algum texto originalmente postado por u_1. As cores representam tópicos que usuários estão falando sobre no instante t. As amostras foram obtidas filtrando os nós com grau entre 50-22000 e as arestas representando os 4 assuntos mais populares. Cada snapshot corresponde a 1 dia de granularidade de tempo.

Tendo então, uma ideia da semântica da rede, uma visualização de amostras dessa rede e algumas estatísticas de métricas de centralidade como um todo, é possível extrair diversas informações. Alguns dos principais conhecimentos são listados a seguir:

- A centralidade closeness está relacionada com a visibilidade de um nó na rede. É a capacidade que um nó tem de atingir os demais de maneira rápida. Portanto, um alto closeness significa uma boa capacidade de espalhar informações. Nesse contexto, é possível então identificar 3 tipos de usuários: consumidores, produtores e consumidores&produtores. Consumidores são aqueles que na maioria das vezes apenas retweetam, não publicando nenhum novo conteúdo. Geralmente, têm um baixo closeness. Produtores são aqueles que sempre estão publicando novos conteúdos, com closeness médio. Por fim, os consumidores&produtores têm um alto grau de atividade na rede, tweetando e retweetando o tempo todo. Esse tipo de usuário possui os mais altos valores de closeness.

- A predominância dos assuntos mais comentados na rede varia a cada dia. Tal efeito é intuitivo, já que reflete o dia-a-dia do contexto de notícias: a cada nova manchete, novos comentários e interações relacionados a ela surgem, trazendo um caráter natural de evolução dos assuntos predominantes na rede.

- É possível identificar quais os tópicos favoritos de cada usuário. Se um determinado usuário sempre interage na rede com o mesmo assunto, já foi possível extrair um conhecimento personalizado em relação àquele usuário.

1.6.2. Rede Social de CDRs

Call Detail Records (CDRs) são registros de ligações telefônicas em empresas de telecomunicações. Em geral, as principais informações contidas em um CDR são: o número

que originou a chamada (número de A), o número que recebeu a chamada (número de B), o instante que a chamada ocorreu e a duração. Uma rede social de CDRs é aquela em que os nós são os números de telefone e as arestas são dirigidas indicando que um número ligou para o outro. Esse estudo de caso foi extraído do trabalho [25]. A Tabela 1.3 sumariza as estatísticas da rede em questão.

Tabela 1.3. Estatísticas da rede social de CDRs.

Descrição da Rede	
Domínio	Ligações telefônicas de uma empresa de telecom
Intervalo de tempo	31 dias
Granularidade do processamento	1 dia
Característica do processamento	*streaming* (fluxo contínuo)
Topologia da rede	
# ligações	386,492,749
# números de telefone	11,916,442

Nesse estudo, o foco está no volume de dados sendo processado e na escolha da abordagem de processamento em *streams* para extrair conhecimento temporal da rede. As Figuras 1.13(a) e 1.13(b) descrevem algumas das características principais da rede em termos de evolução.

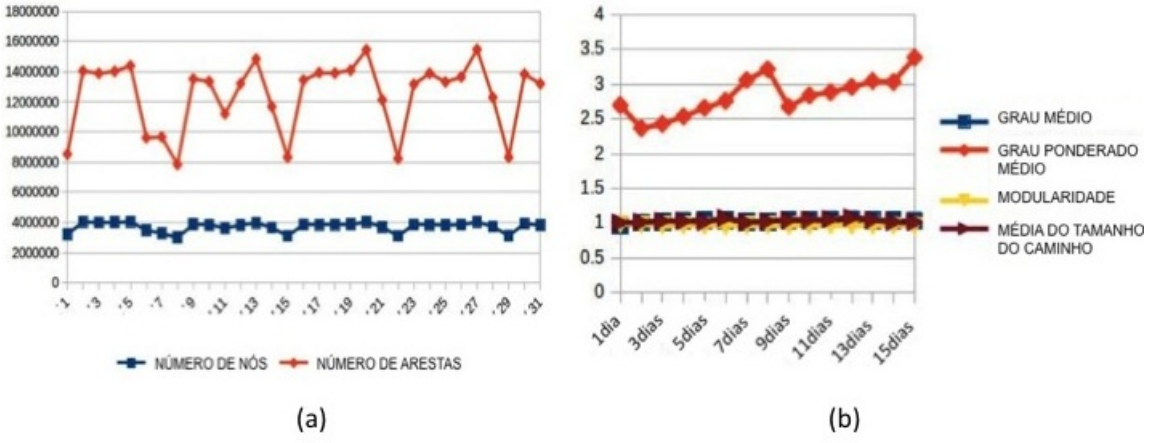


Figura 1.13. Traduzida de [25]. Evolução da rede de CDRs. (a) Evolução dos nós e arestas ao longo de 31 dias. (b) Evolução de métricas de centralidade e modularidade globais da rede.

Quanto à interpretação, é possível enumerar o seguinte conhecimento extraído:

- Existe um padrão de ligações telefônicas ao longo do mês. A cada intervalo de 7 dias o número de ligações diminui indicando os dias de domingo.
- As métricas de rede testadas se mantêm constantes ao longo do período analisado, indicando que a operadora de telefonia conseguiu manter naquele mês os seus clientes e nada de diferente ocorreu para fazer tais clientes mudarem seus padrões de comunicação.

Nesse cenário, então, além da visualização da rede (não ilustrada aqui por restrições de espaço) foi utilizada a estratégia de coletar a cada dia estatísticas da rede para entender sua evolução estrutural. Logo, é uma análise puramente estrutural, diferente da rede do Twitter em que o conteúdo da rede (tópicos) foram utilizados para extração do conhecimento. Perceba que uma entrega de um cientista de dados em rede para a empresa deve ter sempre como foco análises que possam gerar valor ao negócio de telecom.

1.6.3. Rede Social de Músicas

Esse estudo de caso descreve os resultados obtidos no trabalho [19]. A rede social nesse cenário foi obtida a partir de dados da plataforma de músicas Last.fm. Os nós são usuários da plataforma e as arestas são relações de seguidores/seguidos assim como no Twitter. O contexto é então, uma rede de amigos orientada por gostos musicais, onde cada aresta possui a indicação do instante em que a amizade foi criada. As estatísticas estão na Tabela 1.4.

Tabela 1.4. Estatísticas da rede social de músicas.

Descrição da Rede	
Domínio	Relações de amizade na plataforma de músicas Last.fm
Intervalo de tempo	1/1/2002 - 31/12/2011
Granularidade do processamento	1 dia
Topologia da rede	
# usuários	71,000
# arestas	285,241

Utilizando a mesma estratégia de análise descrita no caso de uso da rede de CDRs, uma primeira análise exploratória pode ser descrita com gráficos que mostram a evolução do comportamento da rede. As Figuras 1.14(a) e 1.14(b) descrevem o comportamento da rede em questão.

Assim como no contexto da rede do Twitter, essa análise foi feita acrescentando aos nós (usuários) informações sobre seus comportamentos – quais as músicas, playlists e artistas foram ouvidos. O objetivo desse estudo, retirado do trabalho [19], foi validar o nível de influência que uma rede de amizades pode exercer sobre os gostos musicais. Retomando à discussão proposta sobre a sequência básica de passos para análise de redes sociais temporais, nesse estudo de caso foram observadas as estatísticas, evolução do comportamento da rede e métricas. Só depois de entender bem a rede que se tem em mãos é que análises mais profundas e específicas – como o grau de influência dos amigos nas escolhas musicais – devem começar.

1.7. Considerações Finais

Neste capítulo foram discutidas estratégias para extração de conhecimento em redes sociais temporais. Tais redes representam a evolução das interações entre entidades ao longo do tempo, podendo ser aplicadas em contextos como Twitter, redes de ligações telefônicas e redes de gostos musicais. Primeiro foram definidos diversos conceitos por trás da ideia temporal de análise de redes. Tais conceitos mostram que as métricas de centralidade

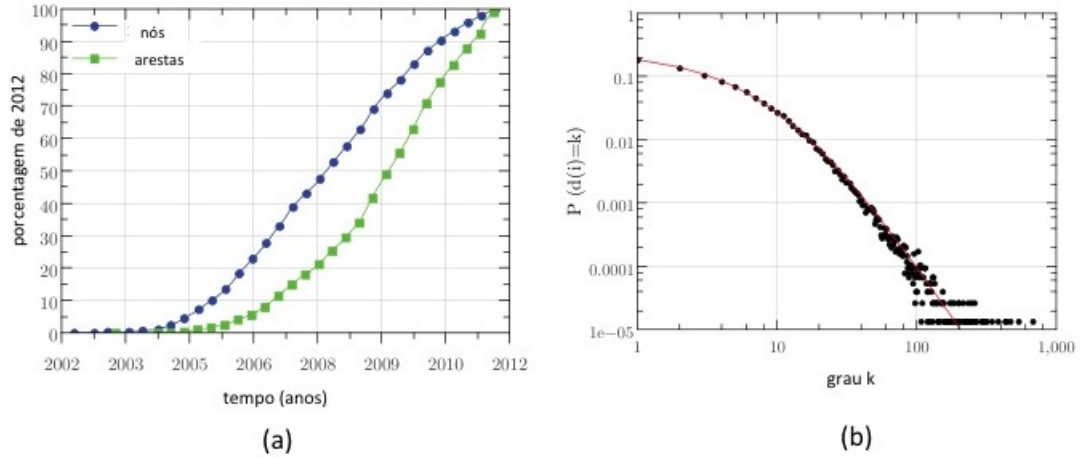


Figura 1.14. Traduzida de [19]. Evolução da rede social de amizades da plataforma Last.fm. (a) Evolução dos nós e arestas ao longo dos anos. (b) Distribuição dos graus dos nós da rede.

baseadas em caminhos devem ser revisitadas e repensadas no contexto temporal. Em seguida, foram discutidas estratégias para processamento de redes que evoluem à medida que o tempo passa, variando entre estratégias para redes que evoluem lentamente e para redes com atualização constante, em fluxos de arestas. Uma vez apresentados os conceitos, em caráter prático, foram mostradas técnicas de visualização que podem fornecer *insights* no momento da análise da rede, bem como ferramentas que auxiliam tais atividades. Por fim, os conceitos, técnicas e ferramentas foram aplicadas em três estudos de caso que ilustram como deve ser uma análise para descoberta de conhecimento em redes sociais temporais.

Referências

[1] Adedoyin-Olowe, M., Gaber, M.M., Stahl, F.: A Survey of Data Mining Techniques for Social Media Analysis. Journal of Data Mining & Digital Humanities 2014 (Jun 2014), http://jdmdh.episciences.org/18

[2] Ahmed, N.K., Neville, J., Kompella, R.: Network sampling: From static to streaming graphs. ACM Trans. Knowl. Discov. Data 8(2), 7:1–7:56 (Jun 2013), http://doi.acm.org/10.1145/2601438

[3] Bach, B., Pietriga, E., Fekete, J.D.: Visualizing dynamic networks with matrix cubes. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 877–886. CHI '14, ACM, New York, NY, USA (2014)

[4] Bastian, M., Heymann, S., Jacomy, M.: Gephi: An open source software for exploring and manipulating networks (2009)

[5] Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D.U.: Complex networks: Structure and dynamics. Physics Reports 424(4–5), 175–308 (2006)

[6] Carley, K.M., Pfeffer, J.: Dynamic network analysis (dna) and ora. Advances in Design for Cross-Cultural Activities Part I p. 265Ð274 (2012)

[7] Costa, L.F., Rodrigues, F., Travieso, G., Villas Boas, P.R.: Characterization of complex networks: a survey of measurements. Advances in Physics 56(1), 167–242 (2007)

[8] Franca, T.C., de Faria, F.F., Rangel, F.M., de Farias, C.M., Oliveira, J.: Big social data: Principios sobre coleta, tratamento e analise de dados sociais. In: Anais do SBBD. pp. 1–40 (2014)

[9] Guha, S., McGregor, A.: Graph synopses, sketches, and streams: A survey. Proc. VLDB Endow. 5(12), 2030–2031 (Aug 2012), http://dx.doi.org/10.14778/2367502.2367570

[10] Holme, P.: Analyzing temporal networks in social media. Proceedings of the IEEE 102(12), 1922–1933 (2014)

[11] Holme, P., Saramaki, J.: Temporal networks. Physics Reports 519(3), 97–125 (2012)

[12] Huhtamäki, J.: Visualizing co-authorship networks for actionable insights: Action design research experiment. In: Proceedings of the 20th International Academic Mindtrek Conference. pp. 208–215. AcademicMindtrek '16, ACM, New York, NY, USA (2016)

[13] Huisman, M., Duijn, M.A.J.V.: A Reader's Guide to SNA Software. Sage (2011)

[14] Kempe, D., Kleinberg, J., Kumar, A.: Connectivity and inference problems for temporal networks. In: Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing. pp. 504–513. ACM (2000)

[15] Kim, H., Anderson, R.: Temporal node centrality in complex networks. Phys. Rev. E 85, 026107 (Feb 2012)

[16] Linhares, C.D.G., Travençolo, B.A.N., Paiva, J.G.S., Rocha, L.E.C.: Dynetvis: A system for visualization of dynamic networks. In: Proceedings of the Symposium on Applied Computing. pp. 187–194. SAC '17, ACM, New York, NY, USA (2017)

[17] Nicosia, V., Tang, J., Mascolo, C., Musolesi, M., Russo, G., Latora, V.: Temporal Networks, chap. Graph Metrics for Temporal Networks, pp. 15–40. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)

[18] Oezbek, C., Prechelt, L., Thiel, F.: The onion has cancer: Some social network analysis visualizations of open source project communication. In: Proceedings of the 3rd International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development. pp. 5–10. FLOSS '10, ACM, New York, NY, USA (2010)

[19] Pálovics, R., Benczúr, A.A.: Temporal influence over the last.fm social network. In: Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. pp. 486–493. ASONAM '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2492517.2492532

[20] Pan, R.K., Saramäki, J.: Path lengths, correlations, and centrality in temporal networks. Phys. Rev. E 84 (2011)

[21] Pereira, F.S.F., de Amo, S., Gama, J.: Detecting events in evolving social networks through node centrality analysis. Workshop on Large-scale Learning from Data Streams in Evolving Environments co-located with ECML/PKDD (2016)

[22] Pereira, F.S.F., Amo, S., Gama, J.: Evolving centralities in temporal graphs: a twitter network analysis. In: Mobile Data Management (MDM), 2016 17th IEEE International Conference on (2016)

[23] Rossi, R., Gallagher, B., Neville, J., Henderson, K.: Role-dynamics: Fast mining of large dynamic networks. In: Proceedings of the 21st International Conference on World Wide Web. pp. 997–1006. WWW'12 Companion, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2187980.2188234

[24] Srivastava, J.: Data mining for social network analysis. In: 2008 IEEE International Conference on Intelligence and Security Informatics. pp. 23–24. IEEE, Taiwan (June 2008)

[25] Tabassum, S., Gama, J.: Sampling massive streaming call graphs. In: Proceedings of the 2016 ACM Symposium on Applied Computing. pp. 923–928. SAC '16, ACM, New York, NY, USA (2016)

[26] Tang, J., Musolesi, M., Mascolo, C., Latora, V.: Temporal distance metrics for social network analysis. In: Proceedings of the 2nd ACM Workshop on Online Social Networks. pp. 31–36. WOSN '09 (2009)

[27] Viswanath, B., Mislove, A., Cha, M., Gummadi, K.P.: On the evolution of user interaction in facebook. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks (WOSN'09) (August 2009)

[28] Wei, W., Carley, K.M.: Measuring temporal patterns in dynamic social networks. ACM Trans. Knowl. Discov. Data 10(1), 9:1–9:27 (Jul 2015)

[29] Wu, H., Cheng, J., Huang, S., Ke, Y., Lu, Y., Xu, Y.: Path problems in temporal graphs. Proceedings of the VLDB Endowment 7(9), 721–732 (2014)

[30] Zafarani, R., Abbasi, M.A., Liu, H.: Social Media Mining: An Introduction. Cambridge University Press, New York, NY, USA (2014)

1.8. Sobre os Autores

Fabíola Souza Fernandes Pereira. Doutoranda em Ciência da Computação na Universidade Federal de Uberlândia (UFU), com período sanduíche no LIAAD, um grupo pertencente ao INESC TEC, Portugal. Possui graduação (2009) e mestrado (2011) em Ciência da Computação também pela UFU. É autora de artigos peer-reviewed nas áreas de redes temporais, análise de redes sociais e preferências do usuário. Atuou como chair da special session em redes evolutivas (EvoNets) na conferência DSAA'17.

João Gama. É professor associado da Faculdade de Economia, Universidade do Porto. É pesquisador e vice-diretor do LIAAD, um grupo pertencente ao INESC TEC. Obteve o título de Ph.D. pela Universidade do Porto em 2000. Tem trabalhado em diversos projetos nacionais e europeus em sistemas de aprendizado incremental e adaptativo, descoberta de conhecimento ubíquo, aprendizado a partir de dados massivos e em fluxo, etc. É autor de diversos livros em Mineração de Dados e de mais de 250 artigos peer-reviewed nas áreas de aprendizado de máquina, mineração de dados e data streams.

Gina Maira Barbosa de Oliveira. Bolsista de produtividade do CNPq de 2001 a 2017 (PQ-2). Possui graduação em Engenharia Elétrica pela Universidade Federal de Uberlândia (1990), mestrado em Engenharia Eletrônica e Computação pelo Instituto Tecnológico de Aeronáutica (1992) e doutorado em Engenharia Eletrônica e Computação pelo Instituto Tecnológico de Aeronáutica (1999). Pós-doutorado de 07/2013 a 07/2014 na Heriot-Watt University (Edinburgh-Scotland) na área de robótica bio-inspirada. Atualmente é professora associada da Universidade Federal de Uberlândia. Tem experiência na área de Ciência da Computação, atuando principalmente nos seguintes temas: algoritmos genéticos, autômatos celulares, computação evolutiva, computação bio-inspirada, robótica bio-inspirada e inteligência artificial.

mini:2

Capítulo 2

Sports Analytics: Mudando o Jogo

Ígor Barbosa da Costa, Carlos Eduardo Santos Pires e Leandro Balby Marinho

Abstract

In the last decades, researchers have been developing different techniques to understand which factors influence the sporting results and, consequently, what is the role of predictability and randomness in sports. With the evolution of techniques to acquire, store, and process large volumes of information, Sports Analytics has become even more important for discovering new knowledge and transforming the behavior of those involved with the sport. In this chapter we present an introduction to the topic, making a historical contextualization, detailing the types of data used and discussing the process of knowledge discovery for applied research in the domain. In addition, we highlight the relationship between sports data analysis and betting markets. Finally, we present some emerging challenges for the beginning researcher and make our final considerations on this field of research.

Resumo

Nas últimas décadas, pesquisadores vêm desenvolvendo diferentes técnicas para entender quais fatores influenciam os resultados esportivos e, consequentemente, qual o papel da preditibilidade e da aleatoriedade no esporte. Com a evolução das técnicas de aquisição, armazenamento e processamento de grandes volumes de informações, as análises de dados esportivos (Sports Analytics) se tornaram ainda mais importantes para a descoberta de novos conhecimentos e vêm transformando o comportamento daqueles envolvidos com o esporte. Neste capítulo apresentamos uma introdução ao tema, fazendo uma contextualização histórica, detalhando os tipos de dados utilizados e discutindo o processo de descoberta de conhecimento para pesquisas aplicadas no domínio. Além disso, destacamos a relação entre a análise de dados esportivas e os mercados de aposta. Por fim, apresentamos uma série de desafios emergentes para o pesquisador iniciante e fazemos nossas considerações finais sobre esse campo de pesquisa.

2.1. Introdução

No século XX, os esportes passaram por diversas transformações, com destaque para o surgimento das confederações (que cumprem papel regulador) e a profissionalização dos atletas. Além disso, a globalização e a extensão do alcance das mídias sociais permitiram aos fãs acompanhar as façanhas de seus times e atletas diariamente, impulsionando a popularidade de muitos esportes.

Essa crescente popularidade dos esportes permitiu o aumento de investimentos financeiros para clubes e atletas. O esporte profissional se tornou um produto e a indústria esportiva passou a ser um dos maiores mercados de entretenimento no mundo (atualmente, as marcas ligadas ao esporte movimentam cerca 1% de todo o PIB mundial [25]). Os salários dos atletas de destaque dispararam, assim como a cobrança por melhores resultados.

Na constante busca por melhores resultados, a análise de dados esportivos, também conhecida como *Sports Analytics*, ganhou um papel de protagonismo no século XXI. Pode-se definir *Sports Analytics* como a análise de uma coleção de dados históricos que, se realizada adequadamente, pode trazer vantagem competitiva para um time ou atleta.

Em [34], Cokins et al. publicaram um artigo intitulado *"Sports Analytics taxonomy, V.1.0"* que busca classificar as diversas áreas de aplicação da análise de dados no esporte de forma mais abrangente. O trabalho sugere que os ramos da análise esportiva podem ser organizados em três grandes grupos: *Grupo 1*: refere-se ao esporte como competição, ou seja, engloba a análise de equipes, atletas e ligas; *Grupo 2*: refere-se ao esporte como recreação, ou seja, a análise é focada no desempenho e saúde do indivíduo; e *Grupo 3*: tenta superar a incerteza do esporte, ou seja, a análise é voltada para apostas esportivas, jogos de azar, ligas fantasias[1] e jogos eletrônicos.

Na maior parte deste trabalho, trataremos o esporte como competição (*Grupo 1*) e focaremos nas modalidades com interação entre os adversários, como esportes de combate (boxe, judô, MMA, etc.), esportes de campo e taco (beisebol, críquete, etc.), esportes com rede divisória ou parede de rebote (voleibol, tênis, squash, etc.) e esportes de invasão (basquete, futebol, handebol, etc.). A natureza desse tipo de esporte torna a análise de dados uma tarefa mais desafiadora, visto que existe uma diversidade de comportamentos nas estratégias dos times e atletas. Além disso, devido à popularidade dessas modalidades, a quantidade de dados disponíveis para análise está em contínuo crescimento, abrindo novas oportunidades para pesquisa.

A seguir, contextualizaremos *Sports Analytics* na história, destacando o momento em que a análise de dados começou a revolucionar os esportes. Na Seção 2.2, destacaremos um outro domínio impactado pela análise de dados esportivos: o mercado de apostas esportivas (*Grupo 3*). Mostraremos os fundamentos que relacionam *Sports Analytics* e as pesquisas aplicadas em apostas esportivas (*Sports Betting Analytics*). Na Seção 2.3, identificaremos as características dos dados utilizados em *Sports Analytics*, destacando as principais formas de coleta. Além disso, discutiremos como um pesquisador pode ter acesso a essas coleções de dados. Na Seção 2.4, detalharemos os passos da pesquisa apli-

[1] Um tipo de jogo online no qual os participantes escalam equipes imaginárias ou virtuais de jogadores reais de um esporte profissional

cada, discutindo o caminho que o dado percorre até virar uma informação útil, através de exemplos. Na Seção 2.5, apresentaremos os desafios emergentes em *Sports Analytics*. Finalmente, na Seção 2.6, apresentaremos nossas considerações finais a respeito de todo o conteúdo do capítulo.

2.1.1. Contexto Histórico

Apesar de ter recebido maior relevância neste século, a análise de dados esportivos é uma atividade relativamente antiga. Em meados do século XX, alguns pesquisadores já manipulavam dados estatísticos para tentar entender melhor sobre as características dos esportes.

No futebol, Charles Reep, reconhecido como o primeiro analista de dados desta modalidade, criou um sistema notacional para anotar cada lance que ocorria em uma partida. Cada evento do jogo recebia uma categorização detalhada. Por exemplo, para cada passe efetuado, era registrada a posição no campo onde o passe foi originado e finalizado, assim como a distância, a direção e a altura do passe. Os dados coletados durante 15 anos foram base para o artigo científico intitulado *"Skill and Chance in Association Football"* publicado em 1968 [47].

No referido artigo, Reep, juntamente com o estatístico Bernard Benjamin, buscavam entender se os dados coletados podiam revelar padrões previsíveis do esporte. Dentre as descobertas, podemos destacar a definição de que o futebol era um processo estocástico: um chute em cada oito termina em gol, mas era difícil determinar qual deles. Descobriram também que o futebol é um jogo de alternância, pois a maioria das jogadas termina após zero ou um passe completo, ou seja, em uma partida, a posse da bola é trocada, em média, quatrocentas vezes. Além disso, os autores demonstraram que 30% de todas as bolas recuperadas na grande área do adversário se transformam em finalizações ao gol e que metade dos gols eram resultado dessas mesmas bolas recuperadas.

No beisebol, Bill James desafiou a análise tradicional do jogo ao demonstrar que as estatísticas utilizadas pelas equipes para avaliar o desempenho dos atletas estavam equivocadas. Por exemplo, na análise tradicional, a média de rebatidas de um jogador era considerada um indicador importante para definir qual rebatedor deveria iniciar uma partida. Bill, por sua vez, definia que a importância de um jogador estava no modo como ele contribui para as vitórias e não nas suas estatísticas brutas. Ele afirmava que as médias de ataque não significariam nada se o rebatedor não pontuasse. Ideias como esta fizeram com que Bill publicasse uma coleção de "novas estatísticas", denominada *Sabermetrics* [4], para avaliação de atletas de beisebol.

No basquete, Dean Oliver foi um dos pioneiros. Inspirado no *Sabermetrics*, Dean começou a realizar análises semelhantes durante a década de 1980. Buscando quantificar melhor a contribuição dos jogadores para a equipe, criou estatísticas para avaliar o desempenho do time em relação a quantos pontos marcava ou sofria a cada cem posses de bola. Os estudos deram origem ao *APBRmetrics - Association for Professional Basketball Research Metrics* [1].

Entretanto, apesar de pesquisas como as de Reep, James e Oliver, que analisavam o esporte com um certo rigor científico, a análise de dados estatísticos não parecia ter

grande relevância para as tomadas de decisão dos gestores, as quais eram guiadas, principalmente, pelo conhecimento empírico de especialistas do esporte. Foi apenas no início do século XXI, com a história de sucesso do *Oakland Athletics* [21] na MLB[2], que a análise de dados começou a causar uma verdadeira revolução e passou a receber a devida atenção de todos os envolvidos no esporte.

2.1.2. A Revolução dos Dados

Em 2002, o investimento financeiro cada vez maior em algumas franquias[3] estava tornando o beisebol relativamente previsível. As franquias de maior orçamento venciam os campeonatos, enquanto as franquias de baixo orçamento naturalmente ficavam nas últimas colocações e fora dos *play-offs*[4]. Tradicionalmente, as franquias de beisebol trabalhavam da mesma forma. Todas tinham uma equipe de olheiros que decidia quais atletas deveriam ser contratados ou recrutados para a temporada seguinte. Até que, Billy Beane, gerente geral e ex-jogador do *Oakland Athletics*, decidiu que, se ele não conseguia competir com as demais franquias no aspecto financeiro (sua franquia tinha o segundo menor orçamento da MLB), precisava encontrar uma outra vantagem competitiva. Dessa forma, auxiliado pelo economista Paul Depodesta, Beane foi buscar no *Sabermetrics* de James, a solução para nivelar a competição [43].

Beane e Depodeseta montaram o time para temporada 2002 fazendo uma análise baseada apenas nas estatísticas de desempenho do *Sabermetrics*, ignorando aspectos relevantes para os olheiros, tais como idade avançada, porte físico e até mesmo o modo desajeitado de jogar. O resultado foi um time com jogadores de baixo custo, cujas estatísticas denotavam pontos fortes a serem explorados.

A campanha foi surpreendente. O desacreditado time bateu o recorde de 20 vitórias consecutivas da conferência e se classificou para os *play-offs*. Não alcançou o título, mas, no ano seguinte, a inovação foi tema do *best-seller "Moneyball - A arte de ganhar um jogo injusto"* [43]. O sucesso do clube e do livro foram refletidos imediatamente na mudança de ideologia dos grandes clubes que viam seus jogadores com altíssimos salários renderem menos de que aqueles que ganhavam menos.

Desde então, o *Sabermetrics* passou a ser fundamental na seleção dos elencos de todos os clubes da MLB. O tradicional e rico, *Boston Red Sox*, foi além e contratou o próprio Bill James para trabalhar para o clube (onde permanece até hoje). O clube conseguiu em 2004 sagrar-se campeão após 86 anos de jejum, além de ter repetido o feito em 2007 e 2013.

Moneyball mudou não só o Beisebol, como também todos os esportes. Analistas e pesquisadores de outras modalidades passaram a adotar abordagens científicas similares para avaliar o conhecimento empírico relacionado aos esportes. No futebol, por exemplo, Chris Anderson e David Sally publicaram o livro *"Os números do jogo"* [30], compilando uma série de pesquisas que buscavam trazer evidências contrárias às crenças tradicionais do jogo. Dentre as discussões apresentadas nesse trabalho, podemos destacar: a ideia de que o escanteio não deveria ser cobrado para a grande área, a análise de que a demissão de

[2]MLB - Major League Baseball

[3]Nos Estados Unidos, as equipes pertencem a empresas habitualmente chamadas de franquias

[4]Sistema de disputa eliminatório que decide os finalistas de um campeonato

treinadores não melhora o desempenho dos times e a discussão sobre porque um jogador mais fraco (elo fraco) tem mais relevância (negativa) para o resultado do que os melhores jogadores do time (elo forte), ou seja, porque nenhum time é melhor que seu pior jogador.

Se no campo científico os cientistas seguem buscando encontrar padrões no esporte ou fazer predições, no campo esportivo os clubes estão cada vez mais interessados na descoberta de conhecimento através da análise de dados, visando obter vantagem competitiva. Entretanto, além da perspectiva esportiva, o interesse pela criação de modelos preditivos a partir de dados esportivos também tem sido impulsionado pelo crescimento dos mercados de apostas. A seguir, discutiremos a respeito desse domínio.

2.2. Apostas Esportivas: Motivação e Fundamentos

A democratização da Internet impulsionou mundialmente um segmento de mercado que está avaliado atualmente em mais de 3 trilhões de dólares [25]. Estamos falando do mercado de apostas esportivas. Este segmento já representa 37% do mercado de jogos de azar. Embora não exista regulamentação para essa prática em algumas partes do mundo (incluindo o Brasil), as casas de apostas online, hospedadas em lugares onde o jogo é regulamentado, permitem que pessoas de qualquer lugar do mundo possam realizar apostas pela Internet. Dessa forma, existe um grande número de informações disponíveis para análise na própria Web.

Nesta seção, apresentaremos uma fundamentação sobre apostas esportivas, visando dar ao pesquisador o entendimento sobre o significado dos dados desse domínio. Essa fundamentação é importante, pois como veremos na Seção 2.3.1.2, dados do mercado de apostas podem servir como dados de contexto em *Sports Analytics*, da mesma forma que dados esportivos de outras fontes podem servir para resolver desafios das pesquisas aplicadas em apostas esportivas (detalhados na Seção 2.5), também conhecida como *Sports Betting Analytics*.

2.2.1. Conceitos Fundamentais

O mercado de apostas possui dois ramos diferentes: as casas de apostas (*bookmakers market*) e as bolsas de apostas (em inglês, *betting exchange market*). As casas de apostas são as formas tradicionais de realizar apostas. Elas funcionam como reguladoras do mercado, oferecendo diversas oportunidades de apostas para o público. Nesse caso, podemos dizer que o apostador realiza a aposta "contra" uma casa de apostas. Já no outro ramo, o das bolsas de apostas, os apostadores apostam contra outros apostadores através de uma operadora que intermedia a negociação.

O conceito de aposta está estritamente ligado ao conceito de *odd*. De uma forma geral, as *odds* são usadas para definir quanto um apostador receberá se fizer uma aposta bem sucedida. Por essa razão, elas são frequentemente definidas como o "preço" pago por uma aposta premiada. As *odds* são representadas de diferentes formas pelas casas de apostas. Nesta seção, usaremos as *odds* decimais (*european odds*), o formato mais conhecido e utilizado. Outros formatos incluem as *odds* fracionárias (*fractional odds*) e as *odds* americanas (*american odds*).

Pare exemplificar, analisaremos as *odds* decimais para a luta de boxe entre Floyd

Mayweather e Conor McGregor (ver Figura 2.1), realizada em agosto de 2017. Cada resultado possível da luta possui uma *odd* associada e as mesmas nos permitem fazer as seguintes interpretações:

Figura 2.1. *Odds* oferecidas pelo *Sportingbet.com* para a luta entre Floyd Mayweather e Conor McGregor

- Se o apostador fizer uma aposta de $1 em Floyd e ele for o vencedor, o apostador receberá de volta $1.20;
- Se o apostador fizer uma aposta de $1 em Conor e ele for o vencedor, o apostador receberá de volta $4.33;
- Se o apostador fizer uma aposta de $1 no empate e a luta terminar empatada, o apostador receberá de volta $34.00;

Formalmente, considerando o valor apostado como *va* e a *odd* decimal oferecida pela casa de apostas para um determinado resultado como *odd*, podemos definir o potencial valor de retorno *vr* para uma aposta, como mostrado na Equação 1:

$$vr = va * odd \tag{1}$$

Nessa equação, podemos observar que o valor apostado está incluído no potencial valor de retorno, ou seja, para definir o lucro de uma aposta precisamos subtrair uma unidade do valor da *odd* apostada. Formalmente, podemos definir esse *lucro_potencial* como na Equação 2:

$$lucro\_potencial = va * (odd - 1) \tag{2}$$

Uma vez entendido que o valor da *odd* está associado ao retorno que o apostador poderá receber, podemos compreender como as casas de apostas definem o valor da *odd* a ser ofertada. Esse cálculo está diretamente ligado as chances de um determinado evento ocorrer, ou seja, as probabilidades.

2.2.2. *Odds* vs. Probabilidades

Em diversas áreas, é comum que a representação da probabilidade de um determinado evento ocorrer seja dada em termos percentuais. Por exemplo, se jogarmos uma moeda honesta para cima, ela tem 50% de chance de virar cara e 50% de virar coroa. Se jogarmos um dado de seis lados, cada lado tem aproximadamente 16,6% de chance de ocorrer. Conceitualmente, as somas das probabilidades de todos os eventos possíveis deve totalizar 100%.

No mercado de apostas, as *odds* são calculadas a partir dessas probabilidades. Entretanto, não podemos afirmar que essas *odds* refletem exatamente as chances de um determinado evento acontecer. Para perceber essa sutil diferença, precisamos entender o conceito de probabilidade implícita (*implied probability*).

Podemos definir que a probabilidade implícita é inversamente proporcional a *odd*, ou seja, dado um possível resultado x, podemos calcular sua probabilidade implícita *imp*, como na Equação 3:

$$imp_x = \frac{1}{odd_x} \tag{3}$$

Por exemplo, calculando as probabilidades implícitas para os resultados possíveis da luta entre Mayweather e McGregor (ver Figura 2.1), teremos:

- Vitória de Mayweather: $imp_{(Mayweather)} = \frac{1}{1,20} = 83,33\%$
- Vitória de McGregor: $imp_{(McGregor)} = \frac{1}{4,33} = 23,09\%$
- Empate: $imp_{(empate)} = \frac{1}{34,00} = 2,94\%$

Realizando a soma total dessas probabilidades, encontramos o valor 109,36%, que é diferente do valor esperado numa distribuição de probabilidade convencional, que seria 100%. Na prática, as casas de apostas nunca ofertarão *odds* nas quais as somas de suas probabilidade implícitas seja exatamente 100% (na Seção 2.2.3 explicaremos os motivos). Mesmo assim, essa probabilidade é fundamental para que o apostador avalie se o valor esperado (*expected value*) da aposta é positivo. Uma aposta tem "valor esperado positivo"(+*EV*) quando as probabilidades implícitas das *odds* oferecidas pelas casas de apostas são inferiores às probabilidades estimadas pelo apostador. Em outras palavras, se no exemplo da Figura 2.1, o apostador acreditar que McGregor possui 30% de chances de vencer, o apostador parece ter uma boa oportunidade para apostar, dado que a casa de apostas acredita que essa chance é de apenas 23,09%.

2.2.3. O Lucro das Casas de Apostas e o Balanceamento das *Odds*

Na Seção 2.2.2, entendemos que as somas das probabilidades implícitas das *odds* de um evento não totaliza 100% como esperado em um evento justo. Entretanto, é a partir dessa diferença, denominada *overround*, que vem o lucro das casas de apostas. O *overround* é um dos motivos que fazem as casas de apostas serem lucrativas em longo prazo. Elas têm, implicitamente, uma vantagem sobre os apostadores, que varia geralmente de 5% a 12% (no exemplo da Figura 2.1, o valor é de 9,36%). Para melhor entendimento dessa vantagem, vamos voltar ao exemplo da luta de boxe.

Se a casa de apostas tiver feito uma análise correta sobre as chances de cada lutador no evento, isso deverá se refletir no comportamento dos apostadores. Em outras palavras, a distribuição dos valores recebidos pela casa de apostas (em apostas) será semelhante à distribuição das probabilidades oferecidas por ela. Dessa forma, a cada \$109.36 recebidos pela casa de apostas, ela espera que \$83.33 estejam em apostas para Floyd,

$23.09 para Conor e $2.94 para o empate. Nesse cenário ideal, ao final do evento, a casa de apostas poderia ter os seguintes resultados:

- Em caso de vitória de Mayweather, a casa de apostas precisaria pagar $1.2 para cada $1 apostado. Sendo assim, como a casa recebeu $83.33 para esse resultado, pagaria $99.99 aos apostadores, obtendo um lucro de $9.37 sobre o total recebido ($109,36);
- Em caso de vitória de McGregor, a casa de apostas precisaria pagar $4.33 para cada $1 apostado. Nesse caso, como recebeu $23.09, pagaria $99.98 e lucraria $9.38;
- Em caso de empate, a casa precisaria pagar $34.00 para cada $1 recebido. Como recebeu apenas $2.94, pagaria $99.96 e lucraria $9.40.

Observando o cenário apresentado, podemos afirmar que a casa de apostas lucraria independente do resultado do evento. Mas, e se a distribuição dos valores apostados não estiver dentro do esperado? Isso significa que os apostadores, de certa forma, estão discordando das chances oferecidas pela casa. Esse é um dos motivos que pode fazer a casa de apostas recalcular suas probabilidades no decorrer do tempo, numa ação chamada de "balanceamento de *odds*".

Sabendo que as *odds* são publicadas muitos dias (ou meses) antes do início do evento, o balanceamento de *odds* pode ser realizado a qualquer momento, seja durante o evento ou até antes de ele iniciar. Apesar de ser menos comum, o balanceamento antes do evento é realizado para refletir uma nova informação (por exemplo, a contusão de um atleta importante antes de uma partida) ou para se ajustar à percepção do mercado. Entretanto, é durante o evento que existe uma grande variação, visto que as chances podem mudar a cada novo acontecimento (detalharemos isso na Seção 2.2.4). No exemplo da Figura 2.1, é natural que, se no início da luta de boxe, McGregor começar acertando bons golpes, as expectativas mudem e consequentemente as *odds* também.

Por fim, podemos concluir que as casas de apostas não estão necessariamente preocupadas em acertar o resultado final de um evento, mas, sim, em oferecer *odds* que reflitam a expectativa do mercado em relação às chances para cada resultado possível. Para isso, as casas vão estar sempre ajustando suas *odds* (conservando o *overround*) para manter o volume de apostas recebido balanceado e continuar garantindo lucro, independente do resultado do evento.

2.2.4. Casas de Apostas vs. Bolsas de Apostas

Uma vez entendidos os fundamentos que definem uma *odd*, podemos discutir como funcionam os dois ramos do mercado de apostas: casas de apostas e bolsas de apostas.

No ramo de casas de apostas, o apostador realiza a aposta em um determinado resultado, antes ou durante o evento, e aguarda o fim deste para saber se acertou. Em caso de acerto, a casa paga ao apostador de acordo com a *odd* negociada. Em caso de erro, o dinheiro do apostador é lucrado pela casa de apostas. Dentre as principais casas de apostas do mundo estão a Bet365 [13], William Hill [29], Bet-at-Home [12], Bwin [16], Rivalo [24] e Pinnacle [22].

Na bolsa de apostas, a dinâmica é um pouco diferente. Um apostador só consegue realizar uma aposta, caso exista outra pessoa que esteja disposta a apostar contra. Por exemplo, se na luta entre Mayweather e McGregor, um apostador decidir apostar a favor de McGregor, então precisa existir alguém que concorde em apostar contra McGregor (ou seja, em Mayweather ou empate). Existem diversas empresas que intermediam esse tipo de negociação, entre elas se destacam a Betfair.com [15] e a Betdaq.com [14]

A primeira diferença entre esses dois mercados está no valor das *odds* oferecidas. Como as apostas na bolsa são feitas entre pessoas, não existe a necessidade do *overround*, ou seja, as *odds* negociadas são geralmente maiores. Entretanto, as operadoras da bolsa cobram uma taxa de corretagem que varia de 2,5% a 7% sobre a aposta vencedora. Dessa forma, uma *odd* maior na bolsa (do que nas casas de apostas) não significa obrigatoriamente um lucro potencial maior. Para observarmos essas diferenças, vamos comparar as *odds* oferecidas na Betfair para a mesma luta do exemplo anterior, entre Mayweather e McGregor (ver Figura 2.2).

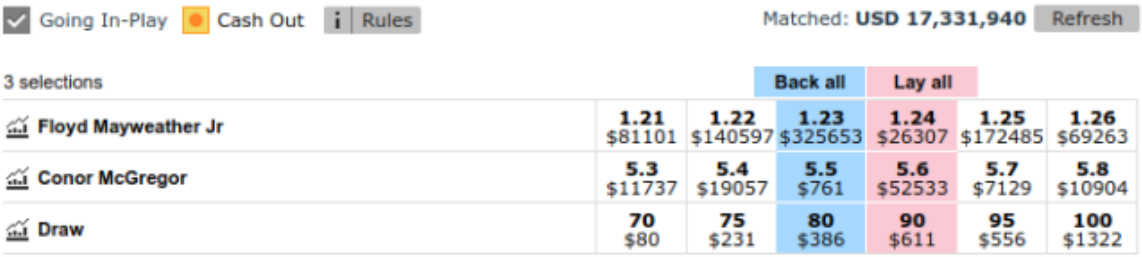


Figura 2.2. *Odds* oferecidas pela *Betfair.com* para a luta entre Floyd Mayweather e Conor McGregor

Ao calcular as probabilidades implícitas de cada *odd*, temos:

- Vitória de Mayweather: $imp_{(Mayweather)} = \frac{1}{1{,}23} = 81,30\%$
- Vitória de McGregor: $imp_{(McGregor)} = \frac{1}{5{,}50} = 18,18\%$
- Empate: $imp_{(empate)} = \frac{1}{80{,}00} = 1,25\%$

Somando as probabilidades da distribuição, encontramos um total de 100,73%, demonstrando um *overround* próximo a zero. Dessa forma, uma aposta de $1 em Mayweather que fosse vencedora, traria um retorno de $1.23. Todavia, se aplicada a taxa máxima de corretagem (7%), o lucro final seria próximo de $0.14 e inferior aos $0.20 que seriam lucrados, caso a aposta fosse feita na casa de apostas. Isso confirma que uma *odd* de 1.20 na casa de apostas pode ser mais lucrativa que uma *odd* de 1.23 encontrada na bolsa de apostas, caso a taxa de corretagem não seja baixa. No caso das *odds* oferecidas para McGregor e para o empate, podemos afirmar que, mesmo com uma taxa alta de corretagem (7%), elas oferecem lucro maior do que nas casas de apostas.

Uma outra diferença entre os ramos, ainda mais significativa, está na dinâmica para encerramento das apostas. Como mencionado anteriormente, é comum que, nas casas de apostas, as pessoas esperem até o final do evento para saber se a aposta teve sucesso,

ou não. Esse tipo de apostador é conhecido como *punter* e no Brasil acabou se tornando também adjetivo para uma forma de aposta. Dessa forma, uma "aposta punter" é aquela em que o apostador aguarda até o final do evento apostado. A bolsa de apostas, apesar de receber diversas apostas *punters*, permite um outro tipo de negociação, conhecida como *trading*. O *trading* esportivo é semelhante ao *trading* das bolsas de valores. O apostador trata a *odd* como o preço de um "ativo", o qual pode comprar (fazer a aposta) e vender segundos ou minutos depois de ter comprado (encerrar a aposta). Nesse caso, a variação positiva ou negativa da *odd* é quem define o lucro ou o prejuízo do apostador.

Assim como na bolsa de valores, existem diversas estratégias para fazer *trading*: *Scalping*, *Swing Trader*, *Dutching*, *Bookmaking*, outre outras. Nesta seção, apenas fizemos uma breve introdução sobre a natureza desse mercado. Em [18], [20], [19] é possível encontrar um extenso material sobre como operar nesse segmento e as possibilidades de apostas.

Do ponto de vista da pesquisa aplicada, alguns desafios serão apresentados na Seção 2.5. Entretanto, antes de discuti-los, precisamos entender as relações entre os dados coletados das casas de apostas e os dados do domínio esportivo (que serão detalhados na Seção a seguir).

2.3. Coleta de Dados

Nos últimos anos, com a crescente relevância de *Sports Analytics* para a indústria esportiva, as tecnologias para coleta automática de dados passaram a ter um grande potencial científico e comercial. Impulsionados pela evolução das tecnologias de armazenamento e processamento de grandes volumes de informações, os novos dispositivos de coleta estão trazendo um desafio ainda maior para os analistas de dados.

Na prática, as equipes estão conseguindo obter coleções heterogêneas de dados (em inglês, *datasets*) relacionadas ao desempenho dos atletas e adversários. Esses *datasets* estão cada vez maiores, com informações que antes eram relativamente impossíveis de serem catalogadas, como, por exemplo, o detalhamento da movimentação dos atletas em campo. A seguir, detalharemos os dados utilizados em *Sports Analytics*, destacando as principais formas de aquisição, de acordo com a classificação de Stein et al. [49]. Em seguida, descreveremos como um pesquisador pode ter acesso a esses dados para iniciar uma pesquisa.

2.3.1. A Origem dos Dados

A aquisição de dados para análise no esporte pode ser realizada de diversas formas. Nos esportes com interação entre os adversários, é comum que os dados sejam extraídos a partir de gravações de vídeos e sensores. Essas informações coletadas podem ser enriquecidas por dados relacionados ao "contexto", como a localização de uma disputa, informações meteorológicas, horário do evento, etc. Informações de contexto também podem ser obtidas a partir de publicações de redes sociais (textos, imagens, *feeds* de vídeo) ou casas de apostas, que refletem a sensação do público em relação aos acontecimentos do evento esportivo. Dessa forma, iremos dividir os tipos de dados sob dois aspectos: os dados extraídos a partir de vídeos e sensores e os dados de contexto.

2.3.1.1. Dados de Vídeos e Sensores

As gravações de vídeos são comuns nos esportes modernos, seja pela perspectiva da mídia (com diversas câmeras espalhadas por estádios e arenas), como também pela perspectiva das equipes que realizam as gravações por conta própria. De uma forma geral, as gravações podem ser consideradas uma das principais fontes de extração de dados, permitindo a indexação de informações referentes a movimentação dos atletas, eventos do jogo e dados descritivos (estatísticos). Empresas como a *STATS* [26] e *Opta Sports* [7] oferecem serviços para catalogar essas informações a partir de suas próprias gravações.

Outra possibilidade para extração desses dados são os sensores colocados diretamente nos jogadores (*wearables*) ou objetos de jogo (bolas, linhas, alvos, etc.). A maioria dos esportes já permitem que as equipes utilizem esses dispositivos, inclusive durante os eventos. Empresas como *Adidas* [10], *Catapult* [17], *Whoop* [28] e a brasileira *One Sports* [6], desenvolveram dispostivos que além de capturar a movimentação dos atletas, rastreiam suas características biométricas.

Dados de Movimentação

Os dados de movimentação, também conhecidos como dados espaço-temporais, descrevem onde um jogador ou objeto está localizado em um momento específico. Essas localizações são geralmente registradas por pares de coordenadas (x,y) e, às vezes, também pela coordenada z.

Os principais clubes do mundo já estão adotando em seus campos de treinamento e arenas sistemas como o *SportVU* da *Stats*. O *SportVU* é um sistema que integra múltiplas câmeras para rastrear as coordenadas de todos os jogadores e da bola (25 vezes por segundo), gerando mais de 3 milhões de dados por treino ou jogo.

Dados de Eventos

Podemos definir evento como um acontecimento relevante de uma disputa esportiva. Uma disputa pode ser descrita como uma sequência ordenada de eventos. A análise automática de vídeo já consegue realizar a detecção de diversos eventos relevantes. Entretanto, também é possível realizar essa catalogação através de notação manual. Na prática, a notação manual pode ter problema de precisão, enquanto que o reconhecimento automático pode ter problemas com a produção de resultados falso-positivos. De forma geral, podemos distinguir os eventos em duas categorias:

- *Eventos baseados em regras*: ocorrem de acordo com as regras de cada modalidade. No futebol, por exemplo, temos os laterais, os escanteios, os gols, etc.;
- *Eventos baseados na interação de atletas*: acontecem a partir da interação do atleta com a bola ou com o adversário. No futebol, por exemplo, temos os passes, os chutes a gol, os cruzamentos, etc.

Dados Descritivos

O desempenho de jogadores e times pode ser caracterizado a partir de dados descritivos. Os dados descritivos incluem tudo o que pode ser contado ou medido durante as

competições. Existe uma grande disponibilidade desse tipo de dado, que inclui resultados históricos, classificações de ligas, registros de carreiras, sumarização de dados de eventos, entre outros.

Historicamente, as estatísticas descritivas são as fontes mais utilizadas para fins analíticos, uma vez que os dados gerados automaticamente por gravações de vídeos são tecnologias relativamente novas.

2.3.1.2. Dados de Contexto

Dados relacionados ao contexto podem ser importantes para algumas análises. Tais dados podem incluir fatores ambientais, tais como a localização do jogo, temperatura ambiente e características do gramado. Fatores dessa natureza, podem, por exemplo, ser utilizados para estimar a qualidade do campo de jogo ou possíveis efeitos sobre o desempenho de um jogador [49].

Além disso, com a popularização das redes sociais, as percepções do público sobre os jogos também podem ser consideradas dados relevantes para as análises. Muitos desses serviços fornecem APIs para coleta seletiva dessas informações, que podem ser realizadas antes, durante e após as competições. Nesse mesmo sentido, esse sentimento coletivo também pode ser observado através do comportamento dos mercados de apostas esportivas (como vimos na Seção 2.2). As casas de apostas disponibilizam uma grande massa de dados que pode ser analisada.

Por fim, outros fatores externos, como notícias de jornais sobre as competições ou dados de documentos Web também podem ser consideradas importantes fontes de dados.

2.3.2. Acesso aos Dados

No tópico anterior, discutimos as características dos dados coletados para *Sports Analytics*. Uma vez compreendida a natureza desses dados, precisamos entender que, na prática, uma parte dessas coleções de dados são confidenciais e acessadas apenas pelos analistas das próprias empresas (que coletam) ou dos clubes (que contratam o serviço). Dessa forma, ainda não existem grandes *datasets* públicos com *dados de movimentação* para a comunidade científica em geral, pois as empresas só disponibilizam esses *datasets* de forma comercial e com um custo bastante elevado.

Por outro lado, existem *datasets* com *dados de eventos* e *dados descritivos* que podem ser acessados a partir de diversas fontes na Internet. Uma lista de bases públicas com dados dessa natureza pode ser encontrada em [42].

Os *dados de contexto*, dependendo da natureza, também podem ser acessados na Internet. Dados de redes sociais ou de casas de apostas, por exemplo, podem ser acessados através de APIs públicas. Uma lista de APIs para redes sociais pode ser encontrada em [9].

Em geral, os dados públicos estão disponíveis de forma estruturada em bases de dados (que podem ser acessadas através de APIs) ou planilhas (que podem ser baixadas por requisição HTTP). Entretanto, em alguns casos, os dados podem estar disponíveis apenas em formato semi-estruturado (por exemplo, embutidos em páginas HTML). Nesse

caso, é comum a realização de duas tarefas em sequência, visando a estruturação do *dataset*: rastreamento (Web crawling) e a raspagem (Web scraping).

O rastreamento é a tarefa de navegar pela Internet de forma automatizada para indexar páginas nas quais os dados desejados estão embutidos. Para realização desta tarefa são desenvolvidos ou utilizados algoritmos conhecidos como *web crawlers* ou *bots*. Existe uma grande número de bibliotecas para esse fim como, por exemplo, a *requests* [23], escrita em *Python*.

A raspagem é a tarefa de extrair informações específicas de documentos Web. Após o rastreamento do documento, analisa-se o mesmo visando compreender sua estrutura. Em seguida, escreve-se um algoritmo (*scraper*) para extrair apenas os dados desejados. Existem diversas bibliotecas para esse fim como, por exemplo, a *BeautifulSoup* [11], escrita em *Python*.

2.4. *Sports Analytics*: Pesquisa Aplicada e Desafios Emergentes

Na Seção 2.3, apresentamos as características dos dados utilizados em *Sports Analytics* e observamos que, com o contínuo crescimento dos dados disponíveis, o uso de métodos computacionais se tornou indispensável para alcançar novos e melhores resultados.

As pesquisas de análise de dados esportivos usando abordagens computacionais são comumente associadas a termos como Inteligência Artificial, Mineração de Dados e Aprendizagem de Máquina. Embora exista uma correlação entre todos esses termos (em alguns casos, podem ser considerados até sinônimos), cada um representa uma perspectiva diferente na modelagem dos dados.

Numa perspectiva geral, *Sports Analytics* é todo o processo de descoberta, interpretação e comunicação de novos conhecimentos obtidos a partir da modelagem de dados esportivos. Numa perspectiva tradicional, modelos podem ser obtidos diretamente a partir de análises estatísticas ou formulações matemáticas. Entretanto, na perspectiva da Ciência da Computação, essas abordagens estatísticas e matemáticas são aplicadas através de técnicas de mineração de dados. Além da estatística, as técnicas de mineração de dados podem utilizar métodos da inteligência artificial. Atualmente, a maioria dos métodos de sucesso de inteligência artificial utilizam aprendizagem de máquina (*machine learning*).

Nesta seção, detalharemos os passos para realização de uma pesquisa aplicada em *Sports Analytics*, destacando técnicas e exemplos que permitam ao pesquisador iniciante ter um ponto de partida para o desenvolvimento de pesquisas na área.

2.4.1. Processo de Descoberta de Conhecimento

Na Ciência da Computação, o processo que compreende todo o ciclo que o dado percorre até virar uma informação útil é conhecido como Processo de Descoberta de Conhecimento em Bases de Dados (*em inglês, Knowledge Discovery in Databases - KDD*). O KDD pode ser definido como um processo não trivial, de extração de informações, previamente desconhecidas e potencialmente úteis a partir de um conjunto de dados [37]. Dessa forma, qualquer pesquisa aplicada em *Sports Analytics* segue implicitamente esse processo.

O KDD contém uma série de passos (ver Figura 2.3) e, por ser um processo interativo, permite que o pesquisador possa intervir no fluxo das atividades, retornando a

passos anteriores quando necessário. A seguir, discutiremos como esse processo deve ser adequado à natureza do objetivo de pesquisa.

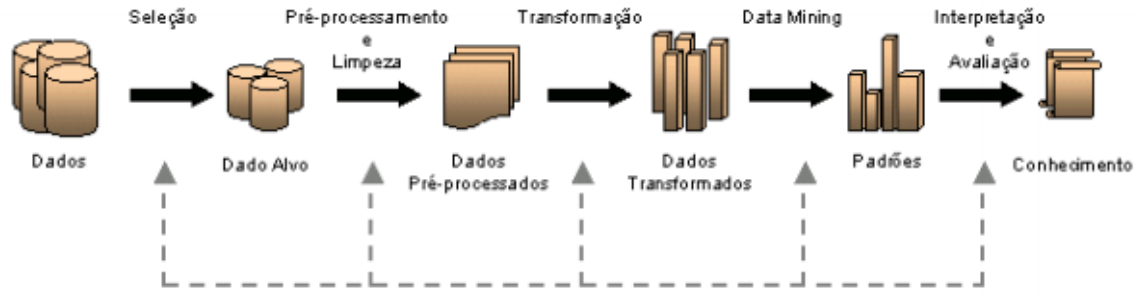


Figura 2.3. Processo de KDD (adaptado de [37])

2.4.1.1. Natureza do Objetivo de Pesquisa

De uma forma superficial, podemos dizer que existem dois objetivos gerais na pesquisa aplicada em *Sports Analytics*: detectar padrões ou realizar predições. Na prática, o processo acabará sendo o mesmo, pois para realizar predições é necessário o reconhecimento dos padrões.

Por exemplo, a detecção de padrões é um objetivo comum na análise de desempenho dos atletas. Os analistas tentam identificar os pontos fortes e fracos de suas equipes ou das equipes adversárias. Em outras palavras, eles buscam entender quais dados podem explicar a derrota ou vitória do seu time. Uma vez entendidos esses dados, os treinadores podem adaptar os treinos para tentar melhorar o desempenho dos atletas. Analogamente, podemos perceber que, se descobrirmos quais dados foram importantes no passado para determinar os resultados das disputas, podemos naturalmente fazer predições para os próximos eventos.

Entretanto, mesmo a modelagem podendo ser semelhante para ambos os objetivos, é preciso ter uma ideia clara sobre a finalidade da pesquisa, antes de iniciar a modelagem. Atualmente, os métodos de mineração de dados podem ser classificados em métodos de "caixa preta"ou métodos de "caixa branca". De uma forma geral, os métodos de caixa preta utilizam abordagens computacionais complexas e tendem a ser mais acurados. Por outro lado, esses métodos geralmente dão pouca informação sobre quais dados foram relevantes para se chegar a determinado resultado. Já os resultados dos métodos caixa-brancas são fáceis de serem interpretados, ao mesmo tempo que podem não ser tão acurados quanto os de caixa-preta.

Dessa forma, reanalisando nosso exemplo anterior, para um analista que deseja entender quais dados influenciaram um determinado resultado, a escolha de métodos caixa-preta pode não trazer informações tão significantes. Por outro lado, para um analista que deseja investir no mercado de apostas, talvez seja mais importante ter um modelo mais acurado, do que entender quais dados podem estar influenciando os resultados das disputas.

2.4.1.2. Seleção de Dados

Uma vez entendida a natureza do objetivo da pesquisa, precisamos decidir quais conjuntos de dados (*datasets*) estão disponíveis e podem ser relevantes para a modelagem do problema. Os *datasets* são compostos por variáveis e registros. As variáveis são conhecidas também como características (*features*), fatores ou atributos, enquanto os registros podem ser denotados como casos, objetos ou observações.

Por exemplo, se desejamos fazer um modelo para prever os resultados de um determinado campeonato, é natural precisarmos de dados históricos para que possamos encontrar variáveis que ajudem nosso modelo a prever corretamente. Considerando que temos disponíveis os resultados desse campeonato nos últimos cinco anos, então cada jogo será um *registro* e cada informação referente ao jogo será uma *variável* (nome do time, gols marcados, data, local, etc.).

Huang & Chang [41], por exemplo, utilizaram diversas variáveis (gols marcados, gols sofridos, chutes pra fora, chutes no alvo, pênaltis, faltas sofridas, faltas cometidas, cartões amarelos, cartões vermelhos, posse de bola, etc.) para fazer predições na Copa do Mundo de Futebol realizada na Alemanha, em 2006.

2.4.2. Pré-processamento e Limpeza de Dados

A seleção de um *dataset* é, geralmente, sucedida por uma análise exploratória, na qual podemos analisar os dados disponíveis e identificar alguns comportamentos iniciais. Para essa análise exploratória, podemos usar estatística descritiva como gráficos descritivos ou descrições tabulares e paramétricas. Este é um passo muito importante a ser realizado junto com um especialista no domínio do esporte em questão, pois através dessas análises podem surgir *insights* importantes para o pré-processamento dos dados.

Após essa análise exploratória, podemos ter mais segurança para realizar a limpeza e o pré-processamento dos dados. O passo de **limpeza de dados** é realizado através de uma série de tarefas, que incluem os tratamentos de valores ausentes, *outliers*, dados inconsistentes e dados duplicados. Já o passo de **pré-processamento** envolve tarefas que serão muito importantes para alcançar o objetivo de pesquisa, visto que, em pesquisas aplicadas, o sucesso dos métodos de mineração de dados dependem principalmente do trabalho de engenharia dos dados (*feature engineering*). Este passo envolve uma série de tarefas, a citar: agregação, amostragem, redução de dimensionalidade, seleção de subconjunto de características, criação de novos recursos, discretização e binarização de variáveis (detalharemos cada uma destas tarefas nos exemplos a seguir).

Vamos supor que desejamos prever os resultados do Campeonato Brasileiro de Futebol de 2017. Para isso, selecionamos um *dataset* que contém todos os resultados da história do Campeonato Brasileiro (1971-2016), pois acreditamos que essa seja uma boa fonte para tentar criar um modelo preditivo. As variáveis disponíveis no *dataset* estão listadas na Tabela 2.1.

Seguindo o processo do KDD, o primeiro passo é avaliar a qualidade dos nossos dados para fazer a limpeza adequada. Uma lista de atividades possíveis seria:

1. *Dados ausentes*: verificar se todos os placares foram devidamente registrados, ou

Tabela 2.1. Variáveis do *dataset* de resultados do Campeonato Brasileiro

CAMP - Ano do campeonato
NROD - Número da rodada
TIMC - Time da casa
TIMV - Time visitante
GOLC - Gols marcados pelo time da casa
GOLV - Gols marcados pelo time visitante
DATA - Data do jogo
LOCJ - Local do jogo (cidade)

seja, se todos os registros tem valores para os atributos *GOLC* e *GOLV*;

2. *Dados inconsistentes*: verificar se todos os valores para *GOLC* e *GOLV* são números inteiros não-negativos;
3. *Dados duplicados*: verificar se não existem dois registros para o mesmo jogo.

Para cada uma dessas atividades, precisamos tomar uma decisão que poderia resultar na eliminação de alguns registros ou na imputação de valores adequados baseados em algum critério.

2.4.2.1. Amostragem

Após a limpeza dos dados, podemos partir para o pré-processamento dos dados. Começaremos fazendo uma *amostragem*. A amostragem é uma abordagem comumente usada para selecionar um subconjunto de registros a serem analisados. No nosso exemplo, podemos selecionar apenas os registos dos campeonatos a partir de 2006, baseado na informação de que, a partir desse ano, o regulamento do campeonato foi alterado. Essa seria uma amostragem simples. Entretanto, para diferentes propósitos, outras formas de amostragem poderiam ser analisadas. Aoki, Assunção & Melo [31], por exemplo, buscaram medir a influência da sorte em alguns esportes e, para isso, selecionaram uma amostra com dados de 198 campeonatos, incluindo 1.503 temporadas de 84 países diferentes para 4 esportes. É natural que para pesquisas dessa magnitude fosse impossível usar todos os dados da população. Entretanto, foi realizada uma boa amostragem estratificada para avaliação.

2.4.2.2. Criação de Novos Atributos

Em seguida, podemos realizar a *criação de novos atributos*. Esta tarefa depende bastante dos *insights* do pesquisador para resolver o problema proposto. No nosso exemplo, podemos observar que não há um atributo único para revelar qual foi o resultado da partida (vitória do time da casa, empate ou vitória do visitante). Para fazer isso, precisaríamos comparar os gols marcados por cada uma das equipes. Dessa forma, poderíamos começar criando esse atributo que sumariza o resultado (RES). Formalmente, dado um jogo x, podemos detonar RES pela Equação 4.

$$RES_x = \begin{cases} C, & \text{se GOLC}_x > \text{ GOLV}_x \\ V, & \text{se GOLC}_x < \text{ GOLV}_x \\ E, & \text{caso contrário} \end{cases} \quad (4)$$

Agora que representamos a variável do resultado (*RES*), precisamos analisar quais outras variáveis podem ser significativas para fazermos predições para *RES*. No momento, temos apenas as identificações dos times, a data do jogo, o local do jogo e o número da rodada. Esses dados trazem pouca ou nenhuma informação a respeito do desempenho prévio dos times. Dessa forma, a partir dos resultados coletados, podemos criar uma diversidade de variáveis relacionadas ao desempenho dos clubes que podem ser mais significativas para um modelo de predição. Algumas das variáveis que podemos derivar dos registros estão listadas na Tabela 2.2.

Tabela 2.2. Variáveis derivadas do *dataset* de resultados do Campeonato Brasileiro

RES - Resultado do Jogo
GOLSP - Gols Marcados no Campeonato (Gols pró)
GOLSC - Gols Sofridos no Campeonato (Gols contra)
NVIT - Números de Vitórias
NEMP - Número de Empates
NDER - Número de Derrotas
NJOG - Número de Jogos Disputados

2.4.2.3. Redução de Dimensionalidade

Uma outra tarefa importante para a modelagem (seja por questão de tempo de processamento, memória utilizada ou até mesmo eficácia) é a *redução de dimensionalidade* do *dataset*. No nosso exemplo, poderíamos fazer reduções simples como transformar as variáveis GOLSP e GOLSC em uma única variável que represente o saldo de gols, ou ainda transformar as variáveis NVIT, NEMP e NDER em uma única variável que represente o número de pontos marcados. Em cenários mais complexos, algumas técnicas robustas podem ser aplicadas para essa redução. Zhao and Cen [52], por exemplo, demonstraram o uso da análise de componentes principais ou PCA (do inglês, *Principal Component Analysis*) para unir 13 variáveis que influenciam o resultado de uma partida de futebol, em uma única variável. PCA é técnica de álgebra linear para atributos contínuos que deriva novos atributos (componentes principais) que sejam combinações lineares dos atributos principais, sejam ortogonais (perpendiculares entre si) e capturem a quantidade máxima de variações nos dados. Uma outra técnica da álgebra linear também utilizada para redução de dimensionalidade é a decomposição de valor único ou SVD (do inglês, *Singular Value Decomposition*). Detalhes comparativos dessas técnicas podem ser encontrados em [38].

2.4.2.4. Seleção de Variáveis

Uma outra forma de reduzir dimensionalidade é através da *seleção de um subconjunto de variáveis*, também conhecida como *feature selection*. No nosso exemplo, intuitivamente, podemos perceber que o número da rodada ou a data do jogo provavelmente não tem qualquer relevância para as predições dos resultados e, portanto, podem ser retiradas da modelagem. Utilizar *features* que não tem relevância para predição, além de aumentar o custo computacional, pode fazer com que a eficácia dos métodos seja comprometida. Existem três formas de realizar essa seleção. A primeira, é ignorar essa tarefa nesse momento, e deixar que essa seleção ocorra naturalmente dentro do algoritmo de mineração de dados (abordagens internas). A outra é usando alguma abordagem independente nesse momento e apenas as variáveis selecionadas para o algoritmo (abordagem de filtro). Por fim, podemos usar alguns métodos de mineração de dados como uma caixa-preta para que esses métodos indiquem qual o melhor subconjunto de atributos (abordagem de envoltório). Tüfekci [51], por exemplo, comparou técnicas de abordagens de filtro e técnicas de envoltório para reduzir um *dataset* com 70 variáveis, visando realizar predições de resultado para o Campeonato Turco de Futebol. Mais detalhes sobre essas técnicas podem ser vistos em [32].

2.4.2.5. Discretização e Binarização

Alguns algoritmos de mineração de dados requerem que os dados das variáveis estejam em formato categórico. Assim, muitas vezes é necessário transformar uma variável contínua em uma variável discreta (discretização) e tanto as variáveis discretas quanto as contínuas podem ser transformadas em atributos binários (binarização). Em resumo, a binarização pode ser considerada a discretização para duas categorias. Existem diversos métodos para realizar essas tarefas. Constatinuou [36], por exemplo, usou um sistema dinâmico [45] para discretizar variáveis como "força do time", "cansaço"e "motivação". A variável "cansaço", por exemplo, era determinada através da quantidade de dias entre um jogo e outro e, depois de aplicada a discretização, denotava valores como "baixa", "média"e "alta". As técnicas mais utilizadas são discutidas em [40].

2.4.2.6. Agregação

Agregação é a combinação de vários registros em um único. Dessa forma a agregação é importante para mudar a escala (de vários registros para um único) e o escopo (o novo registro representa uma nova informação). Do ponto de vista da técnica de mineração, menos dados significa menos tempo de processamento. No nosso exemplo, se o nosso objetivo de pesquisa fosse prever a quantidade de gols do Campeonato Brasileiro de 2017, poderíamos agregar os registros de todos jogos de um ano de campeonato em um único registro com a soma da quantidade de gols de todos os jogos daquele ano. Dessa forma, limitaríamos a quantidade de registros ao número de temporadas do campeonato. Uma desvantagem da agregação é que podemos perder algum detalhe importante dos dados. No nosso exemplo, perderíamos a informação de qual rodada do campeonato possui a maior quantidade de gols.

2.4.3. Transformação de Dados

O processo de transformação é o último antes da mineração de dados e normalmente é realizado de acordo com a técnica que será utilizada. A transformação é aplicada nos valores de uma determinado variável para que eles sejam reduzidos a uma faixa menor de valores. No nosso problema das predições do Campeonato Brasileiro, poderíamos, por exemplo, transformar a quantidade de gols marcados (GOLP) em média de gols marcados, aplicando uma função simples f (Equação 5) que divide o número de gols feitos pelo número de jogos disputados, para todo registro x.

$$f(x) = \frac{GOLP_x}{NJOG_x} \tag{5}$$

Apesar de serem uma representação da mesma característica, esse tipo de transformação deve ser feita com cuidado, pois altera a natureza dos dados. Entretanto, a redução da variação entre eles torna os dados mais apropriados para algumas técnicas de mineração. É comum a normalização por *min-max*, *z-score* ou escala decimal.

2.4.4. Mineração de Dados

Uma vez obtidos os dados transformados, podemos escolher o método de mineração de dados adequado para atingir o objetivo de pesquisa. Na abordagem computacional os principais métodos de mineração utilizam aprendizagem de máquina. Como detalhamos anteriormente, em *Sports Analytics*, dois objetivos gerais são os mais comuns: a realização de predições e o reconhecimento de padrões e

Para modelagem preditiva destacaremos as técnicas de Regressão e de Classificação, enquanto que para reconhecimento de padrões, destacaremos as técnicas de Agrupamento (*clustering*).

2.4.4.1. Regressão

A regressão é uma técnica de modelagem preditiva na qual a variável dependente (variável alvo) é contínua. Em *Sports Analytics*, podemos usar regressão para problemas como: Quantos pontos um time marcará num jogo de basquete? Quantas horas um tenista vai jogar para vencer um campeonato? Por qual preço Cristiano Ronaldo deve ser negociado?

Formalmente, suponha que X representa um conjunto de dados que contem N observações:

$$X = \{(x_i, y_i) | i = 1, 2, 3, ..., N\} \tag{6}$$

Cada x_i corresponde ao conjunto de atributos da observação de índice i (variáveis explicativas) e y_i corresponde à variável alvo (ou resposta). Em outras palavras, y é o que desejamos prever e x é o conjunto de atributos que pode nos ajudar nessa tarefa.

Sendo assim, podemos dizer que regressão é a tarefa de aprender uma função alvo $\boldsymbol{f}$ que mapeie cada conjunto de atributos x em um valor contínuo y. Se a relação entre as variáveis puder ser descrita por uma função linear, podemos dizer que estamos tratando de uma *regressão linear*; caso não seja descrita por uma função linear (uma

função exponencial ou logarítmica, por exemplo) estaremos tratando de uma *regressão não-linear*.

Quando o conjunto de atributos contém apenas um atributo, estamos tratando de uma *regressão simples*. Um regressão linear simples pode ser observada na Figura 2.4 que trata a previsão da quantidade de finalizações de uma partida a partir da quantidade de escanteios [30].

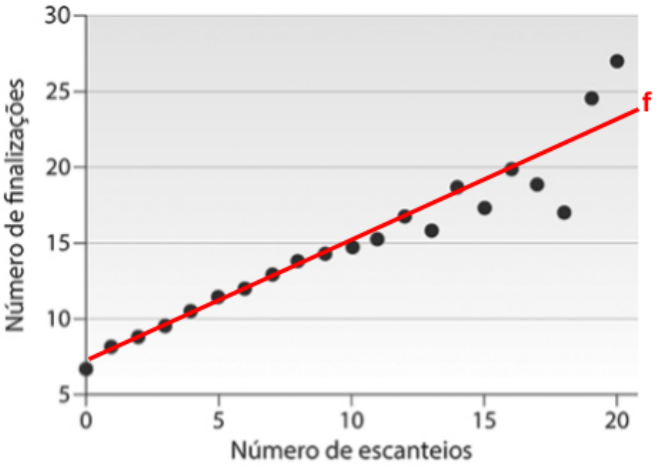


Figura 2.4. Relação entre escanteios e finalizações na *Premier League*, entre 2001-2011 (adaptado de Anderson & Sally [30])

Entretanto, em *Sports Analytics*, é comum que diversos atributos influenciem no valor da variável alvo. No exemplo anterior, poderíamos acrescentar outros atributos (como tempo de posse de bola, número de faltas, etc.) para tentar melhorar a predição da quantidade de finalizações. Nesse caso, quando o conjunto de atributos tem tamanho maior que um, estamos tratando de uma *regressão múltipla*.

Formalmente, a função alvo $\boldsymbol{f}$ de uma regressão linear múltipla pode ser dada pela Equação 7, na qual os valores de $\alpha_1, \alpha_2, ..., \alpha_n$ são os parâmetros a serem estimados na tarefa de regressão e ε é uma variável aleatória desconhecida (erro amostral).

$$f(X) = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + ... + \alpha_n x_n + \varepsilon \tag{7}$$

Em resumo, para se obter a equação estimada, podemos utilizar o método dos mínimos quadrados (MMQ) para obter o menor erro possível. O erro é a representação da soma das diferenças entre o valor real observado e o valor estimado pela regressão. Sendo assim, essa função de erro pode ser denotada, por exemplo, pela soma dos erros quadrados (Equação 9) ou absolutos (Equação 8).

$$Erro\ Absoluto = \sum |y_i - f(x_i)| \tag{8}$$

$$Erro\ Quadrado = \sum (y_i - f(x_i))^2 \tag{9}$$

Em mineração de dados, existem diversas abordagens para estimar os parâmetros da regressão, como Regressão Linear Bayesiana, Regressão por Redes Neurais, Regressão por Árvore de decisão, etc.

2.4.4.2. Agrupamento (*Clustering*)

É uma técnica para agrupar observações segundo algum grau de semelhança, de forma automática. O critério de semelhança é dado por alguma função estatística. O agrupamento é realizado sem intervenção do usuário, sem considerar previamente as características dos grupos e sem o uso de grupos de teste previamente conhecidos para direcionar a classificação.

Os problemas de agrupamento apresentam uma complexidade de ordem exponencial, ou seja, métodos de força bruta, como criar todos os possíveis grupos e escolher a melhor configuração, não são viáveis. Por exemplo, se quisermos agrupar os 100 melhores jogadores de futebol do mundo em 5 grupos, vão existir $5^{100} \approx 10^{70}$ possíveis agrupamentos. Dessa forma, mesmo um computador capaz de testar 109 configurações diferentes por segundo, precisaria de 1.053 anos para terminar a tarefa. Logo, é necessário encontrar uma heurística eficiente que permita resolver o problema [44].

Dois tipos de algoritmos são amplamente utilizados em *Sports Analytics*: os baseados em partição e os baseados em hierarquia. Os primeiros são métodos que buscam agrupar as observações em um número *k* de grupos previamente escolhido, minimizando uma função de custo. Em outras palavras, cada observação será agrupada na classe em que a função de custo é minimizada (ver Figura 2.5). Já os métodos hierárquicos não necessitam que seja definido um número de grupos previamente. Os dados são particionados sucessivamente, produzindo uma representação hierárquica dos agrupamentos (ver Figura 2.6).

Um dos algoritmos de particionamento mais utilizados é o *K-means*, que tem baixa complexidade e grande eficiência computacional em geral. O *K-means* pode ser abstraído em quatro passos:

1. Distribua aleatoriamente *k* pontos (número de clusters) como centros de cluster;
2. Atribua cada observação a um cluster, de forma que a distância da observação ao centro do cluster atribuído seja a menor;
3. Recalcula o centro do cluster usando as observações atribuídas a cada cluster;
4. Repita as etapas 2 e 3 até que as atribuições do cluster não mudem.

Cheng em [27], por exemplo, utilizou *K-means* para classificar os jogadores da NBA em 8 grupos. A rotulagem de grupos foi feita após a tarefa de agrupamento, de acordo com as semelhanças dos jogadores agrupados (ver Figura 2.5). Já Lesmeister [2], utilizou um agrupamento hierárquico, para agrupar 40 *wide receivers*[5] da NFL (ver Figura 2.6).

2.4.4.3. Classificação

É uma técnica de modelagem que mapeia objetos em uma das várias categorias pré-definidas. Na prática, em *Sports Analytics*, podemos usar classificação para fazer pre-

[5]Jogador de posição ofensiva no futebol americano

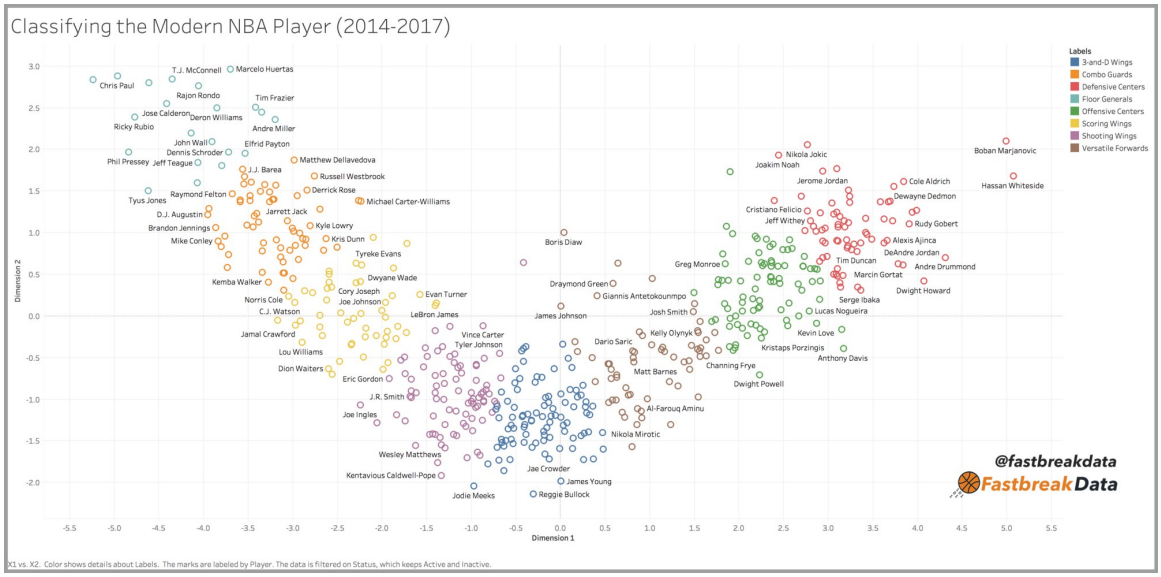


Figura 2.5. Agrupamento particional de jogadores da NBA, entre 2014-2017 (adaptado de Cheng [27])

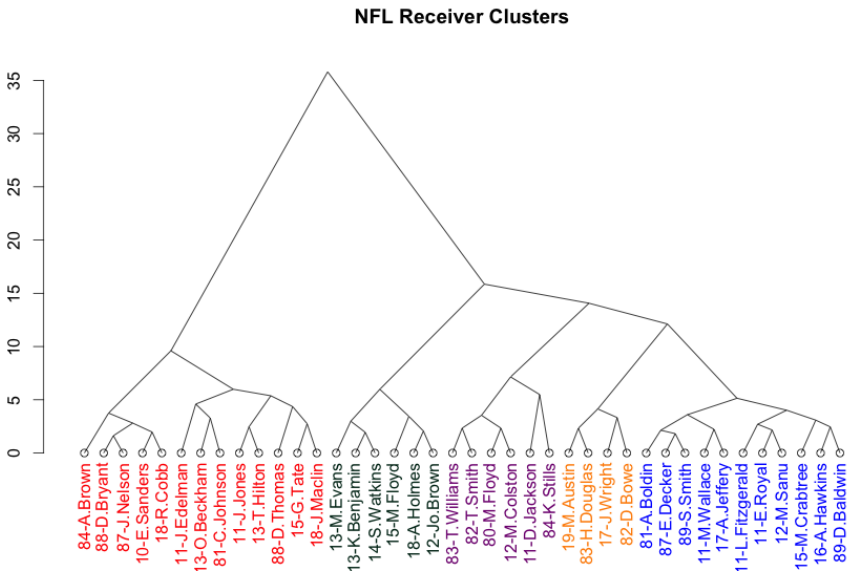


Figura 2.6. Agrupamento hierárquico de 40 *wide receivers* da NFL (adaptado de Lesmeister [27])

dições como, por exemplo: *Quem vencerá um jogo de volêi*? *Em uma partida de futebol, ambos os times marcarão gols*? *Como terminará uma luta de Judô (por pontos*?

A seguir, destacaremos algumas abordagens utilizadas por métodos de classificação em mineração de dados, no escopo de *Sports Analytics*.

Árvores de Decisão

É uma estrutura semelhante a um fluxograma em que cada nó interno representa a avaliação de uma variável, cada ramo representa o resultado do teste e cada nó de folha

representa um rótulo de classe (decisão tomada após o cálculo de todos os atributos). Os caminhos da raiz para a folha representam regras de classificação.

Em [8], por exemplo, o autor utilizou árvores de decisão para prever a posição de um jogador da NBA, a partir de um conjunto de 15 características. A árvore criada para essa classificação pode ser visualizada na Figura 2.7.

Em aprendizagem de máquina, algumas abordagens do estado da arte utilizam múltiplas árvores de decisão. Nesse caso, dois algoritmos bastante utilizados são: Floresta Randômica (*Random Forest*) e *AdaBoost*.

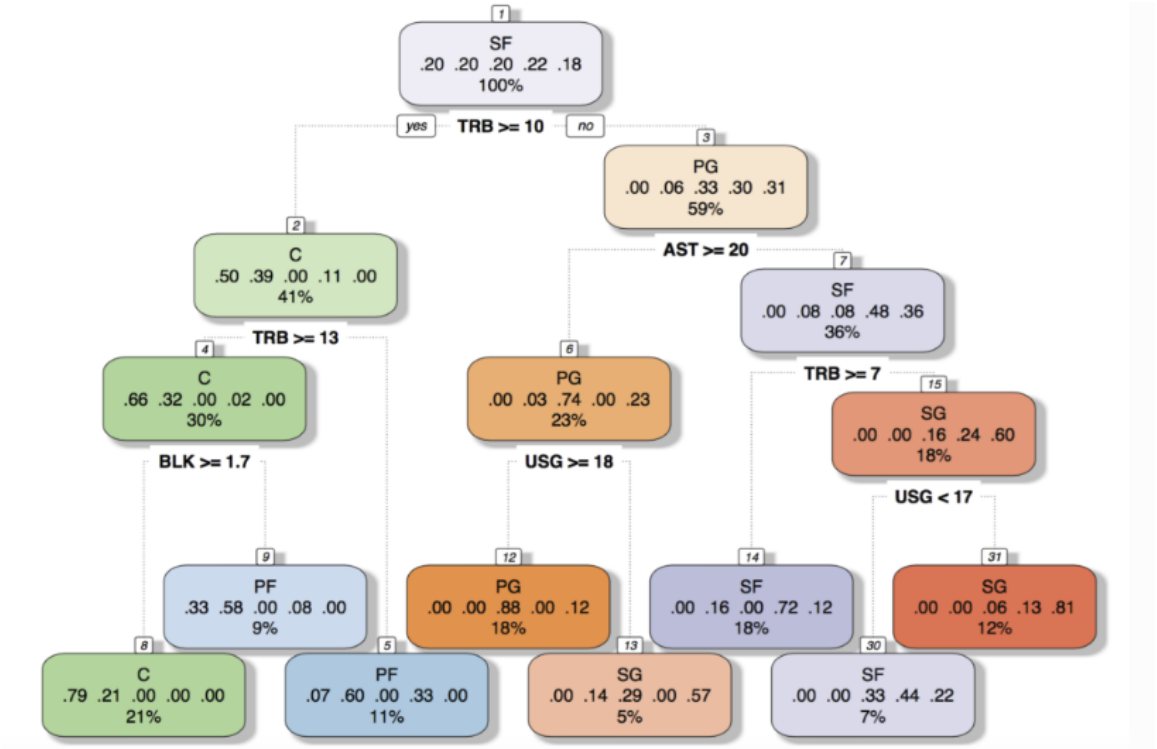


Figura 2.7. Árvore de decisão para classificar jogadores de basquete da NBA por posição (adaptado de Jager [35])

Redes Bayesianas

É um modelo acíclico e probabilístico que representa as variáveis e suas dependências através de um gráfico. Os nós representam as variáveis de um domínio, enquanto os arcos representam as dependências condicionais entre as variáveis. As informações sobre cada nó são dadas através da função de probabilidade que requer um determinado conjunto de valores como entrada e fornece uma distribuição de probabilidade de variáveis como saída.

Constantinou [35] usou Redes Bayesianas para fazer predições de resultados no Campeonato Inglês de Futebol (*Premier League*), através de variáveis objetivas e subjetivas. Uma parte da rede proposta pode ser vista na Figura 2.8.

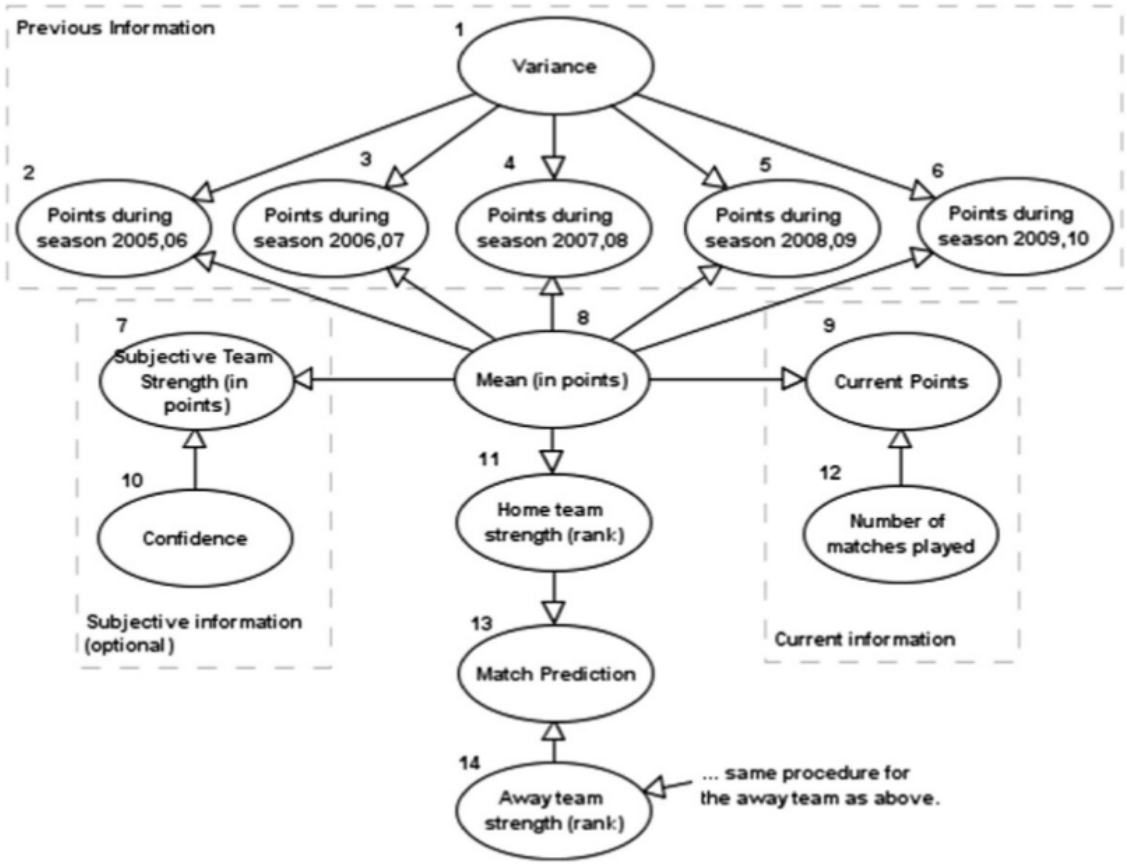


Figura 2.8. Rede Bayesiana para predição de resultados de futebol (adaptado de Constantinou [35])

Redes Neurais

Uma rede neural artificial (ANN) é um modelo computacional baseado em redes neurais biológicas. É uma abordagem de "caixa preta"e, apesar de ser acurada em muitos cenários, não fornece qualquer informação sobre a significância das variáveis independentes.

A técnica consiste em um grupo interconectado de neurônios artificiais (análogos aos do cérebro humano) e processos de informação usando uma abordagem conexionista para computação. Os neurônios dentro da rede trabalham em conjunto (e em paralelo) para produzir uma função de saída. Isso distingue as redes neurais dos programas de computação tradicionais que simplesmente seguem instruções em ordem sequencial. Na maioria dos casos, a ANN é um sistema adaptativo que altera sua estrutura com base em informações que fluem através da rede durante a fase de aprendizagem.

Huang [41], por exemplo, utilizou *multi-layer perceptron* (MLP), um tipo de rede neural, para fazer predições de resultados na Copa do Mundo de Futebol de 2006. A arquitetura utilizada pode ser observada na Figura 2.9.

Máquinas de Vetor de Suporte

A Máquina de Vetor de Suporte (MVS) analisa, para cada observação de conjunto de dados, qual de duas possíveis classes a observação faz parte. Isso faz da MVS um classi-

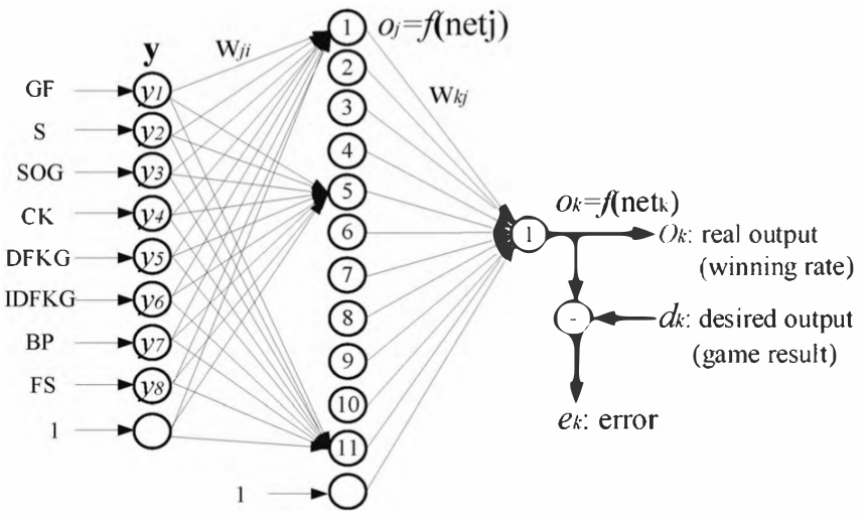


Figura 2.9. Rede Neural para predição de resultados de futebol (adaptado de Huang [35])

ficador linear binário não probabilístico. Em outras palavras, a tarefa da MVS é encontrar uma linha de separação, comumente chamada de hiperplano, entre dados de duas classes. Essa linha busca maximizar a distância entre os pontos mais próximos em relação a cada uma das classes.

Tolbert & Trafalis [50], por exemplo, utilizaram MVS para prever vencedores na MLB. O classificador estimado para esse objetivo pode ser visto na Figura 2.10

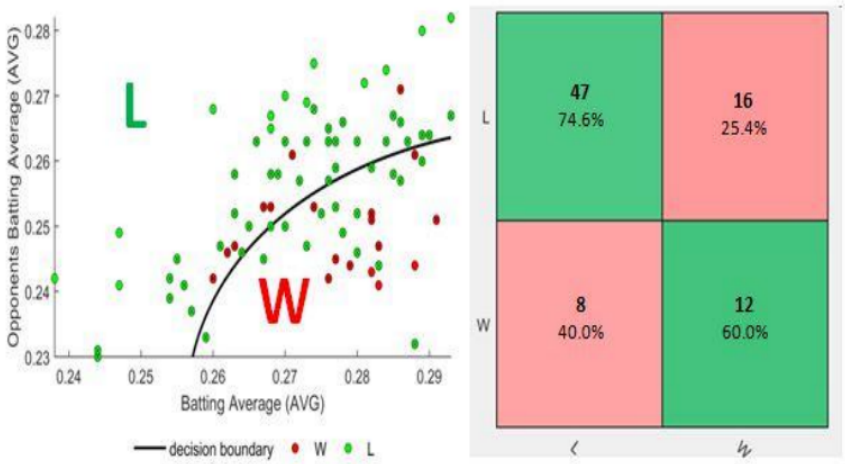


Figura 2.10. Classificador MVS para predição de vencedores no Beisebol (adaptado de Tolbert & Trafalis [50])

2.4.5. Avaliação e Interpretação dos Modelos

A última etapa do KDD é avaliar se o modelo está adequado para o objetivo proposto. Geralmente, a avaliação é parte integrante do processo de desenvolvimento do modelo. Entretanto, avaliar o desempenho do modelo com os dados utilizados para o treinamento não é aceitável na mineração de dados. Dessa forma, existem dois métodos comuns de

avaliação de modelos na mineração de dados: *Hold-Out* e Validação Cruzada (*Cross-Validation*). Para evitar a sobreposição, ambos os métodos utilizam um conjunto de testes (não visto pelo modelo) para avaliar o desempenho do modelo.

No *Hold-Out*, o conjunto de dados é dividido em três subconjuntos:

- **Treinamento**: subconjunto de dados usado para construir modelos preditivos;
- **Validação**: subconjunto de dados utilizado para avaliar o desempenho do modelo construído na fase de treinamento;
- **Teste**: subconjunto (não usado no treinamento) para avaliar o desempenho futuro do modelo. Se um modelo se encaixa no conjunto de treinamento muito melhor do que o conjunto de testes, é provável que este esteja com sobreajuste (*overfitting*).

O conjunto de dados pode ser dividido em quantidades iguais ou não. Geralmente, são selecionados 2/3 dos dados para treinamento e 1/3 para testes.

Quando apenas uma quantidade limitada de dados está disponível para obter uma estimativa do desempenho do modelo, é comum usamos a Validação Cruzada k-fold. Neste tipo de validação, dividimos os dados em k subconjuntos de igual tamanho. Em seguida, um subconjunto é utilizado para testes e os demais são utilizados para estimativa dos parâmetros. Este processo é realizado k vezes, alternando de forma circular o subconjunto de testes. A cada ciclo, calcula-se a acurácia do modelo. Ao término das k interações, calcula-se a acurácia final do modelo sobre os erros encontrados.

A comparação entre modelos pode ser realizada por diversas métricas. Em modelos de regressão, as métricas mais comuns são: Coeficientes de Determinação, Medidas Estatísticas Padrão (Erro Médio, Erro Absoluto Médio e Erro Quadrático Médio) e Medidas Relativas (Erro Percentual, Erro Percentual Médio e Erro Percentual Absoluto Médio).

A comparação entre modelos de classificação também pode ser feita por diferentes métricas ou abordagens como: Taxa de Classificação Incorreta (*misclassification rate*), Matriz de Confusão, F-Measure, Gráficos ROC e Área sob a Curva ROC.

2.5. Desafios Emergentes

Uma vez entendido o detalhamento das pesquisas aplicadas em *Sports Analytics*, podemos listar alguns dos desafios emergentes nesse campo de pesquisa. Do ponto de vista prático, os analistas de dados esportivos têm uma diversidade de desafios para enfrentar, seja no âmbito corporativo (nos clubes ou empresas) ou no âmbito científico.

O grande desafio da pesquisa aplicada está na arte de manipular as variáveis disponíveis para que as técnicas de mineração consigam obter sucesso. Como vimos no processo de KDD, os passos de Pré-Processamento e Transformação são decisivos para que o objetivo seja alcançado. Dessa forma, os desafios emergentes estão relacionados com a natureza dos dados disponíveis. Vejamos:

- **Transformar dados de movimentos em dados de eventos**: fazer essa rotulagem automaticamente tem um alto grau de dificuldade, pois deve ser feita de acordo com

as regras e características de cada esporte. No futebol, por exemplo, a rotulagem de chutes a gol pode ser relativamente trivial, enquanto a rotulagem de uma falta é uma tarefa de alta complexidade. Essas rotulagens podem ser importantes, por exemplo, para gerar os melhores momentos de um jogo, automaticamente.

- **Prever onde os eventos vão acontecer**: os dados também podem ser observados numa perspectiva espacial. Nesse sentido, podemos abstrair quando as coisas aconteceram e nos preocupar apenas com onde aconteceram. Para esse tipo de abordagem, a geometria computacional pode ser o ponto de partida para analisar trajetórias em dados de esportes de interação (ver Figura 2.11). Encontrar padrões nessa sequência também é um tarefa desafiadora. Uma introdução a mineração em dados espaciais pode ser encontrada em [48].

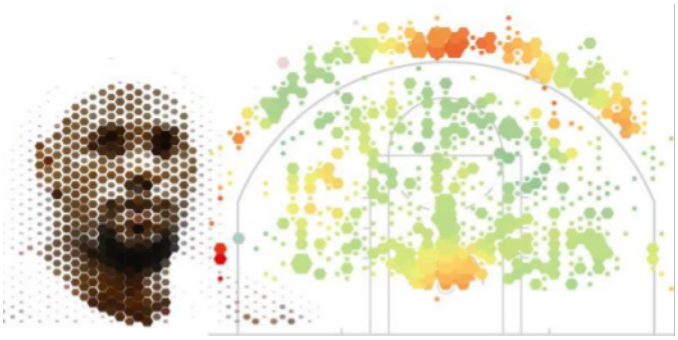


Figura 2.11. Mapa de calor de um jogador de basquete (adaptado de Goldsberry [3])

- **Prever eventos futuros**: os dados podem ser tratados como observações ao longo do tempo, ou seja, como uma série temporal (ver Figura 2.12). Do ponto de vista temporal, podemos abstrair onde as coisas aconteceram e nos preocupar apenas com a sequência de eventos. Dessa forma, encontrar padrões nessas sequências é um tarefa desafiadora. A análise de séries temporais é comum em diversos domínios. Uma lista de técnicas utilizadas pode ser vista em [39].

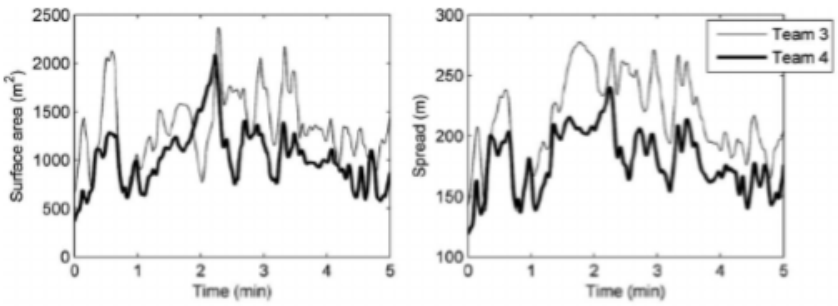


Figura 2.12. Séries temporais do espalhamento dos jogadores e áreas de superfícies cobertas por dois times de futebol em uma disputa (adaptado de Moura et al. [46])

- **Analisar desempenho dos atletas**: a análise de comportamento dos atletas pode ser vista sob a perspectiva individual ou coletiva. Em ambos os casos, a análise dos

dados espaço-temporais (como dados de movimentação) é um dos grandes desafios emergentes na área. Uma introdução sobre o assunto pode ser vista em [33]. Na perspectiva do esporte coletivo, uma abstração para os dados espaço-temporais é interpretá-los como dados de "movimento de grupo". O movimento em grupo também é estudado na biologia e na ciência comportamental. Esses ramos de pesquisa também usam dados de movimentação de pessoas e animais. Atualmente, existe uma diversidade de *datasets* disponíveis para análise [5] com informações dessa natureza. Os pesquisadores acreditam que um dos grandes desafios atuais é encontrar semelhanças e diferenças entre as estratégias das equipes esportivas e o comportamento dos animais em grupo, acreditando que as técnicas de análise desenvolvidas para um domínio possam ser utilizadas no outro [49]. Na perspectiva individual, quantificar o custo-benefício de cada atleta ou encontrar atletas com características semelhantes também são desafios a serem explorados.

- **Prever resultados de jogos, campeonatos ou quantidade de eventos**: a predição de resultados continua sendo um grande desafio, devido à natureza estocástica dos esportes. Nesse contexto, a modelagem de "novas estatísticas"para medição de desempenho das equipes é um desafio permanente. Quantificar a influência da sorte e identificar quais fatores são relevantes para determinar um resultado são sub-tarefas igualmente desafiadoras. Com o crescimento dos mercados de apostas esportivas, diversos outros tipos de predição também passaram a ter relevância, como, por exemplo, quantos escanteios terá um partida de futebol ou quantos gols serão marcados em cada parte do jogo.
- **Resolver desafios específicos de *Sports Betting Analytics***: a análise de dados na perspectiva do mercado financeiro tem uma série de desafios em aberto como, por exemplo: a avaliação de eficiência de mercado (modelos que verifiquem a hipótese de "mercado eficiente", na qual um apostador não consegue obter lucros superiores à média do mercado); a avaliação de estratégias de gestão de banca, difundidas entre os apostadores (*all-in*, Fibonacci, Martingale, Critério de Kelly, valor fixo, etc.); a modelagem de estratégias de *trading* automáticas ou a avaliação da eficiência de outras já utilizadas no mercado financeiro (*Scalping*, *Swing trader*, *Dutching*, etc.); ou ainda, a detecção de fraudes, a partir de apostas suspeitas que possam ter sido originadas a partir de jogos manipulados.

2.6. Considerações Finais

Uma nova era está chegando para o esporte e definitivamente "os dados"são os protagonistas desta revolução. Tudo está sendo mapeado. Jogadores praticam suas atividades com dispositivos que monitoram seus aspectos biométricos. Sistemas de visão computacional estão rastreando todos os passos dos atletas numa granularidade sem precedentes. Na parte administrativa, os clubes identificam o perfil dos seus fãs, descobrindo seu hábitos e analisando seus comentários em redes sociais.

No momento atual, a análise humana ainda é parte da solução. Entretanto, em um futuro próximo, o esporte estará centrado em uma quantidade de dados tão grande que o trabalho de análise não será mais adequado para um humano. O avanço final (que já começou) será a criação de sistemas para o processamento de todas essas informações em

tempo real para que todos os dados sejam transformados em conhecimento, permitindo tomadas de decisão de forma imediata.

O papel do treinador não será extinto, mas definitivamente receberá uma nova conotação. As máquinas serão mais capacitadas que os humanos para avaliar os dados e sugerir mudanças, mas essas mudanças ainda precisarão ser comandadas por pessoas (ao menos, em médio prazo).

Nesse futuro, as máquinas indicarão quais jogadores devem ser contratados e os treinamentos mais eficazes; alertarão quais jogadores estão próximos de se lesionar e como devem ser tratados; avaliarão as táticas da equipe, apontando as falhas e sugerindo as mudanças, em tempo real. Tudo estará, de alguma forma, sendo monitorado minuciosamente.

Os erros humanos de arbitragem também estão com prazo de validade. As regras dos jogos estarão "sob o olhar" de juízes eletrônicos, tornando o esporte mais justo. O mercado de apostas, assim como a bolsa de valores, deverá ser dominada por algoritmos preditivos. Enquanto os jogos de videogame terão times imbatíveis comandados pelas máquinas.

Ao mesmo tempo que esse futuro parece ser inevitável, as oportunidades de pesquisas em análise de dados esportivos se multiplicam. Os atletas continuarão protagonistas, mas os engenheiros e cientistas de dados assumirão papel de destaque nos bastidores. Uma competição paralela e silenciosa que já começou.

Referências

[1] Association for professional basketball research. http://www.apbr.org/. (Acessado em 21/08/2017).

[2] Cluster analysis of the nfl's top wide receivers | r-bloggers. https://www.r-bloggers.com/cluster-analysis-of-the-nfls-top-wide-receivers/. (Acessado em 21/08/2017).

[3] Exploding nba basketball shot heat map analysis - information aesthetics. http://infosthetics.com/archives/2012/06/exploding_nba_basketball_shot_heat_map_analysis.html. (Acessado em 21/08/2017).

[4] A guide to sabermetric research | society for american baseball research. http://sabr.org/sabermetrics. (Acessado em 21/08/2017).

[5] Movebank. https://www.movebank.org/. (Acessado em 21/08/2017).

[6] One sports - a vantagem que gera resultados e campeões. http://www.onesports.com.br/. Acessado em 21/08/2017).

[7] Opta home. http://www.optasports.com/. (Acessado em 21/08/2017).

[8] Predicting nba player positions - nyc data science academy blognyc data science academy blog. http://blog.nycdatascience.com/student-works/predicting-nba-player-positions/. (Accessed on 09/08/2017).

[9] Top 10 social apis: Facebook, twitter and google plus | programmableweb. https://www.programmableweb.com/news/top-10-social-apis-facebook-twitter-and-google-plus/analysis/2015/02/17. (Acessado em 21/08/2017).

[10] Adidas. http://www.adidas.com.br/, 2017. (Acessado em 21/08/2017).

[11] Beautiful soup documentation — beautiful soup 4.4.0 documentation. https://www.crummy.com/software/BeautifulSoup/bs4/doc/, 2017. (Acessado em 21/08/2017).

[12] Bet-at-home.com. https://www.bet-at-home.com/, 08 2017. (Acessado em 21/08/2017).

[13] Bet365.com. https://www.bet365.com/, 08 2017. (Acessado em 21/08/2017).

[14] Betdaq.com. https://www.betdaq.com/, 08 2017. (Acessado em 21/08/2017).

[15] Betfair.com. https://www.betfair.com/exchange/plus/, 08 2017. (Acessado em 21/08/2017).

[16] Bwin.com. https://sports.bwin.com/en/sports, 08 2017. (Acessado em 21/08/2017).

[17] Catapult. http://www.catapultsports.com/, 2017. (Acessado em 21/08/2017).

[18] Clube da aposta • ganhe dinheiro e aprenda como apostar. https://clubedaposta.com/, 2017. (Acessado em 21/08/2017).

[19] Investimento futebol | aprenda a investir e seja um trader esportivo. https://investimentofutebol.com/, 2017. (Acessado em 21/08/2017).

[20] Mercado das apostas - trading esportivo, trader esportivo. http://www.mercadodasapostas.com/, 2017. (Acessado em 21/08/2017).

[21] Oakland athletics. https://www.mlb.com/athletics, 2017. (Acessado em 21/08/2017).

[22] Pinnacle.com. https://www.pinnacle.com/, 08 2017. (Acessado em 21/08/2017).

[23] Requests: Http for humans — requests 2.18.4 documentation. http://docs.python-requests.org/en/master/, 2017. (Acessado em 21/08/2017).

[24] Rivalo.com. https://www.rivalo.com/pt/apostas/, 08 2017. (Acessado em 21/08/2017).

[25] Statista. https://www.statista.com/topics/1740/sports-betting/, 2017. (Acessado em 21/08/2017).

[26] Stats sport vu. https://www.stats.com/sportvu-football/, 2017. (Acessado em 21/08/2017).

[27] Using machine learning to find the 8 types of players in the nba. https://goo.gl/iiMHGp, 2017. (Acessado em 21/08/2017).

[28] Whoop. http://www.whoop.com/, 2017. (Acessado em 21/08/2017).

[29] Williamhill.com. http://sports.williamhill.com/bet/pt, 08 2017. (Acessado em 21/08/2017).

[30] ANDERSON, C., AND SALLY, D. Os números do jogo: Por que tudo o que você sabe sobre futebol está errado. *São Paulo: Paralela* (2013).

[31] AOKI, R., ASSUNCAO, R. M., AND DE MELO, P. O. Luck is hard to beat: The difficulty of sports prediction. *arXiv preprint arXiv:1706.02447* (2017).

[32] CHANDRASHEKAR, G., AND SAHIN, F. A survey on feature selection methods. *Computers & Electrical Engineering 40*, 1 (2014), 16–28.

[33] CHENG, T., HAWORTH, J., ANBAROGLU, B., TANAKSARANOND, G., AND WANG, J. Spatiotemporal data mining. In *Handbook of Regional Science*. Springer, 2014, pp. 1173–1193.

[34] COKINS, G., DEGRANGE, W., CHAMBAL, S., AND WALKER, R. Sports analytics taxonomy, v1.0 - informs. https://www.informs.org/ORMS-Today/Public-Articles/June-Volume-43-Number-3/Sports-analytics-taxonomy-V1.0, 2017. (Acessado em 21/08/2017).

[35] CONSTANTINOU, A. C., FENTON, N. E., AND NEIL, M. pi-football: A bayesian network model for forecasting association football match outcomes. *Knowledge-Based Systems 36* (2012), 322–339.

[36] CONSTANTINOU, A. C., FENTON, N. E., AND NEIL, M. Profiting from an inefficient association football gambling market: Prediction, risk and uncertainty using bayesian networks. *Knowledge-Based Systems 50* (2013), 60–86.

[37] FAYYAD, U. M., PIATETSKY-SHAPIRO, G., SMYTH, P., AND UTHURUSAMY, R. *Advances in knowledge discovery and data mining*, vol. 21. AAAI press Menlo Park, 1996.

[38] FODOR, I. K. A survey of dimension reduction techniques. Tech. rep., Lawrence Livermore National Lab., CA (US), 2002.

[39] FU, T.-C. A review on time series data mining. *Engineering Applications of Artificial Intelligence 24*, 1 (2011), 164–181.

[40] GARCIA, S., LUENGO, J., SÁEZ, J. A., LOPEZ, V., AND HERRERA, F. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Transactions on Knowledge and Data Engineering 25*, 4 (2013), 734–750.

[41] HUANG, K.-Y., AND CHANG, W.-L. A neural network method for prediction of 2006 world cup football game. In *Neural Networks (IJCNN), The 2010 International Joint Conference on* (2010), IEEE, pp. 1–8.

[42] JONES, B. Where to find sports data | tableau public. `https://public.tableau.com/s/blog/2014/03/where-find-sports-data?elq=4d0891dfcfd3415e9b057dd998bf8cc3`, 2014. (Acessado em 21/08/2017).

[43] LEWIS, M. *Moneyball: The art of winning an unfair game*. WW Norton & Company, 2004.

[44] LINDEN, R. Técnicas de agrupamento. *Revista de Sistemas de Informação da FSMA 4* (2009), 18–36.

[45] MARQUEZ, D., NEIL, M., AND FENTON, N. Improved reliability modeling using bayesian networks and dynamic discretization. *Reliability Engineering & System Safety 95*, 4 (2010), 412–425.

[46] MOURA, F. A., MARTINS, L. E. B., ANIDO, R. O., RUFFINO, P. R. C., BARROS, R. M., AND CUNHA, S. A. A spectral analysis of team dynamics and tactics in brazilian football. *Journal of Sports Sciences 31*, 14 (2013), 1568–1577.

[47] REEP, C., AND BENJAMIN, B. Skill and chance in association football. *Journal of the Royal Statistical Society. Series A (General) 131*, 4 (1968), 581–585.

[48] SHEKHAR, S., ZHANG, P., HUANG, Y., AND VATSAVAI, R. R. Spatial data mining.

[49] STEIN, M., JANETZKO, H., SEEBACHER, D., JÄGER, A., NAGEL, M., HÖLSCH, J., KOSUB, S., SCHRECK, T., KEIM, D. A., AND GROSSNIKLAUS, M. How to make sense of team sport data: From acquisition to data modeling and research aspects. *Data 2*, 1 (2017), 2.

[50] TOLBERT, B., AND TRAFALIS, T. Predicting major league baseball championship winners through data mining.

[51] TÜFEKCI, P. Prediction of football match results in turkish super league games. In *Proceedings of the Second International Afro-European Conference for Industrial Advancement AECIA 2015* (2016), Springer, pp. 515–526.

[52] ZHAO, Y., AND CEN, Y. *Data mining applications with R*. Academic Press, 2013.

Sobre os Autores

Ígor Barbosa da Costa é professor de computação do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba (IFPB), campus Campina Grande. Graduado em Ciência da Computação pela Universidade Federal de Campina Grande – UFCG (2006) e Mestre pela Universidade Federal de Pernambuco – UFPE (2010). Tem experiência na área de Ciência da Computação, com ênfase em Bancos de Dados e Desenvolvimento, atuando principalmente nos seguintes temas: Mineração de Dados e Descoberta de Conhecimento (com foco atual em dados esportivos).

Carlos Eduardo Santos Pires concluiu a Graduação em Ciência da Computação, em 1997, pela Universidade Federal de Campina Grande (UFCG) e Mestrado em Informática, em 2000, pela mesma instituição. Em 2009, concluiu o Doutorado na Universidade Federal de Pernambuco (UFPE), tendo realizado Doutorado-Sanduíche na Université de Versailles, na França. Atualmente é Professor Adjunto do Departamento de Sistemas e Computação (DSC), da Universidade Federal de Campina Grande (UFCG). Tem experiência na área de Ciência da Computação, com ênfase em Bancos de Dados, atuando principalmente nos seguintes temas: Qualidade de Dados, Integração de Dados, Descoberta de Conhecimento e Big Data.

Leandro Balby Marinho é Doutor em Ciência da Computação, pela Universidade de Hildesheim, Alemanha, 2010. Mestre em Engenharia Elétrica, UFMA, Brasil, 2005. Bacharel em Ciência da Computação, UFMA, Brasil, 2002. Professor do Departamento de Sistemas e Computação da Universidade Federal de Campina Grande (UFCG). Atua como docente, pesquisador e orientador nos cursos de graduação e pós-graduação em ciência da computação. Suas áreas de especialização são: Inteligência Artificial, Mineração de Dados e Recuperação da Informação.

mini:3

Capítulo 3

Como funciona o *Deep Learning*

Moacir A. Ponti[1] e Gabriel B. Paranhos da Costa[1]

Resumo

Métodos de Aprendizado Profundo (Deep Learning) são atualmente o estado-da-arte em muitos problemas possíveis de se resolver via aprendizado de maquina, em particular problemas de classificação. No entanto, ainda há pouco entendimento de como esses métodos funcionam, porque funcionam e quais as limitações envolvidas ao utilizá-los. Nesse capítulo descreveremos em detalhes a transição desde redes rasas (shallow) até as redes profundas (deep), incluindo exemplos em código de como implementá-las, bem como os principais fatores a se levar em consideração ao treinar uma rede profunda. Adicionalmente, iremos introduzir aspectos teóricos que embasam o uso de modelos profundos, e discutir suas limitações.

Abstract

Deep Learning methods are currently the state-of-the-art in many problems which can be tackled via machine learning, in particular classification problems. However there is still lack of understanding on how those methods work, why they work and what are the limitations involved in using them. In this chapter we will describe in detail the transition from shallow to deep networks, include examples of code on how to implement them, as well as the main issues one faces when training a deep network. Afterwards, we introduce some theoretical background behind the use of deep models, and discuss their limitations.

3.1. Introdução

Nos últimos anos, técnicas de Aprendizado Profundo tem revolucionado diversas áreas de aprendizado de máquina, em especial a visão computacional. Isso ocorreu principalmente por dois motivos: a disponibilidade de bases de dados com milhões de imagens [8, 46], e de computadores capazes de reduzir o tempo necessário para realizar o processamento

[1]ICMC — Universidade de São Paulo, São Carlos, SP, Brasil

dessas bases de dados. Antes disso, alguns estudos exploraram a utilização de representações hierárquicas com redes neurais, tais como o Neocognitron de Fukushima [11] e a rede neural para reconhecimento de dígitos de LeCun [28]. Apesar desses métodos serem conhecidos pelas comunidades de Aprendizado de Máquinas e Inteligência Artificial, grande parte dos esforços dos pesquisadores de outras áreas tiveram como foco outras técnicas. Por exemplo, em Visão Computacional abordagens baseadas em características invariantes a escala, *Bag-of-Features*, pirâmides espaciais e abordagens baseadas em dicionários foram bastante utilizadas durante o final da década de 2000. Características baseadas em frequência foram comumente utilizadas pela comunidade de Processamento de Sinais, Áudio e Fala, enquanto métodos de *Bag-of-Words* e suas variantes foram explorados no Processamento de Linguagem Natural.

Talvez um dos principais marcos que atraiu a atenção das diferentes comunidades tenha sido a publicação do artigo e código-fonte que implementa a rede neural convolucional AlexNet [26]. Os resultados apontavam para uma indiscutível performance estatística de tais métodos para realizar classificação de imagem e, a partir desse ponto, o Aprendizado Profundo (do inglês *Deep Learning* (DL)) passou a ser aplicado a diversas áreas, em particular, mas não exclusivamente, as áreas de Visão Computacional, Processamento de Imagens, Computação Gráfica. Redes neurais convolucionais (do inglês *Convolutional Neural Networks* (CNN)), *Deep Belief Networks*, *Restricted Boltzmman Machines* e *Autoencoders* (AE) começaram a aparecer como base para métodos do estado da arte em diversas aplicações. A competição ImageNet [8] teve grande impacto nesse processo, começando uma corrida para encontrar o modelo que seria capaz de superar o atual campeão nesse desafio de classificação de imagens, além de segmentação de imagens, reconhecimento de objetos, entre outras tarefas.

De fato, técnicas de Aprendizado Profundo oferecem atualmente um importante conjunto de métodos para analisar sinais como áudio e fala, conteúdos visuais, incluindo imagens e vídeos, e ainda conteúdo textual. Entretanto, esses métodos incluem diversos modelos, componentes e algoritmos. A variedade de palavras-chave utilizadas nesse contexto faz com que a literatura da área praticamente se torne uma nova linguagem: Mapas de Atributos, Ativação, Campos Receptivos, *dropout*, ReLU, MaxPool, softmax, SGD, Adam, FC, etc. Isso pode fazer com que seja difícil para que um leigo entenda e consiga acompanhar estudos recentes. Além disso, apesar do amplo uso de redes neurais profundas por pesquisadores em busca da resolução de problemas, há ainda uma lacuna no entendimento desses métodos: como eles funcionam, em que situações funcionam e quais suas limitações?

Nesse texto, iremos descrever as Redes Neurais Profundas (*Deep Neural Networks*), e em particular as Redes Convolucionais. Como o foco é na compreensão do funcionamento desse grupo de métodos, outros métodos de Aprendizado Profundo não serão tratados tais como *Deep Belief Networks* (DBN), *Deep Boltzmann Machines* (DBM) e métodos que utilizam Redes Neurais Recorrentes (do inglês *Recurent Neural Networks* (RNN)) e *Reinforcement Learning*. Para maiores detalhes sobre esses métodos, recomenda-se [10, 27, 47] como referências para estudos que utilizam DBNs e DBMs, e [57, 17, 40] para estudos relacionados a RNNs. Para o estudo de Autoencoders, recomendamos [2] e [14], e para o estudo de Redes Geradoras Adversariais (Generative Adversarial Networks, GANs) [15]. Acreditamos que a leitura desse material será de grande ajuda para o

entendimento dos outros métodos citados e não cobertos por esse texto.

A partir da seção seguinte iremos nos referir ao Aprendizado Profundo pelo seu termo em inglês *Deep Learning* (DL), visto que é esse o mais comumente utilizado mesmo na literatura em lingua portuguesa. Utilizaremos ainda as siglas em inglês pelo mesmo motivo, e.g. CNN.

Esse capítulo está organizado da seguinte forma. A Seção 3.2 apresenta definições que serão utilizadas ao longo do texto bem como os pré-requisitos necessários. A seguir, na Seção 3.3 apresentamos de forma suave os conceitos desde o aprendizado de máquina até o aprendizado profundo, por meio do exemplo de classificação de imagens. As Redes Convolucionais (Seção 3.4) são apresentadas, contextualizando suas particularidades. Então, na Seção 3.5 apresentamos as bases teóricas que até o momento explicam o sucesso dos métodos de aprendizado profundo. Finalmente, considerações finais são discutidas na Seção 3.5.1.

3.2. Deep Learning: pré-requisitos e definições

Os pré-requisitos necessários para entender como Deep Learning funciona, incluem conhecimentos básicos de Aprendizado de Máquinas (ML) e Processamento de Imagens (IP), em particular conhecimentos básicos sobre aprendizado supervisionado, classificação, redes neurais Multilayer Perceptron (MLP), aprendizado não-supervisionado, fundamentos de processamento de imagens, representação de imagens, filtragem e convolução. Como esses conceitos estão fora do escopo deste capítulo, recomenda-se [13, 1] como leituras introdutórias. Assume-se também que o leitor está familiarizado com Álgebra Linear e Cálculo, além de Probabilidade e Otimização — uma introdução a esses tópicos pode ser encontrada na Parte 1 do livro-texto de Goodfellow *et al*. sobre Deep Learning [14]. Ainda assim tentaremos descrever os métodos de forma clara o suficiente para que seja possível o entendimento mesmo com rudimentos dos pré-requisitos.

Métodos que utilizam **Deep Learning** buscam descobrir um modelo (e.g. regras, parâmetros) utilizando um conjunto de dados (exemplos) e um método para guiar o aprendizado do modelo a partir desses exemplos. Ao final do processo de aprendizado tem-se uma função capaz de receber por entrada os dados brutos e fornecer como saída uma representação adequada para o problema em questão. Mas como DL se diferencia de ML? Vamos ver dois exemplos.

Exemplo 1 — classificação de imagens : deseja-se receber como entrada uma imagem no formato RGB e produzir como saída as probabilidades dessa imagem pertencer a um conjunto possíveis de classes (e.g. uma função que seja capaz de diferenciar imagens que contém cães, gatos, tartarugas e corujas). Assim queremos aprender uma função como $f(\mathbf{x}) = y$, em que x representa a imagem, e y a classe mais provável para $\mathbf{x}$.

Exemplo 2 — detecção de anomalias em sinais de fala : deseja-se receber como entrada um áudio (sinal) e produzir como saída uma representação desse sinal que permita dizer se a entrada representa um caso normal ou anômalo (e.g. uma função que permita responder para anomalias na fala de uma pessoa que indique alguma enfermidade ou sín-

drome). Assim, a função a ser aprendida estaria no formato $f(\mathbf{x}) = p$, em que p representa a probabilidade de $\mathbf{x}$ ser um áudio anômalo.

Note que a definição e ambos os exemplos são iguais aos de aprendizado de máquina: o Exemplo 1 é um cenário de aprendizado supervisionado, enquanto o Exemplo 2 se trata de um cenário semi-supervisionado (podendo também ser não supervisionado a depender da base de dados).

De ML para DL : a diferença quando se trata de DL é como se aprende a função $f(.)$. De forma geral, métodos não-DL, comumente referidos na literatura como "superficiais" ou "rasos" (o termo em inglês, que utilizaremos, é *shallow*) buscam diretamente por uma única função que possa, a partir de um conjunto de parâmetros, gerar o resultado desejado. Por outro lado, em DL temos métodos que aprendem $f(.)$ por meio da **composições** de funções, i.e.:

$$f(\mathbf{x}) = f_L(\cdots f_2(f_1(\mathbf{x}_1))\cdots)),$$

onde cada função $f_l(.)$ (o índice l se refere comumente a uma "camada", veremos mais a frente o significado desse termo) toma como entrada um vetor de dados $\mathbf{x}_l$ (preparado para a camada l), gerando como saída o próximo vetor $\mathbf{x}_{l+1}$.

Para sermos mais precisos, cada função faz uso de parâmetros para realizar a transformação dos dados de entrada. Iremos denotar o conjunto desses parâmetros (comumente uma matriz) por W_l, relacionado a cada função f_l, e então podemos escrever:

$$f_L(\cdots f_2(f_1(\mathbf{x}_1, W_1); W_2)\cdots), W_L),$$

onde $\mathbf{x}_1$ representa os dados de entrada, cada função tem seu próprio conjunto de parâmetros e sua saída será passada para a próxima função. Na equação acima, temos a composição de L funções, ou L camadas.

Assim uma das ideias centrais em Deep Learning é aprender sucessivas representações dos dados, intermediárias, ou seja, os $\mathbf{x}_l, l = 1\cdots L$ acima. Os algoritmos de DL resolvem o problema de encontrar os parâmetros W diretamente a partir dos dados e definem cada representação como combinações de outras (anteriores) e mais simples [14]. Assim a profundidade (em termos das representações sucessivas) permite aprender uma sequência de funções que transformam vetores mapeando-os de um espaço a outro, até atingir o resultado desejado.

Por isso é de fundamental importância a hierarquia das representações: cada função opera sobre uma entrada gerando uma representação que é então passada para a próxima função. A hipótese em DL é a de que, se tivermos um número suficiente de camadas L, espaços com dimensionalidade alta o suficiente, i.e., o número de parâmetros W em cada função (iremos detalhar o que isso significa nas próximas seções), e dados suficientes para aprender os parâmetros W_l para todo l, então conseguiremos capturar o escopo das relações nos dados originais, encontrando assim a representação mais adequada para a tarefa desejada [6]. Matematicamente, podemos interpretar que essa sequência de transformações separa as múltiplas variedades (do ponto de vista geométrico, do termo inglês

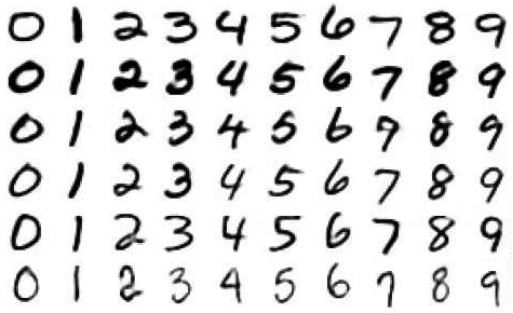


Figura 3.1. Exemplos de imagens das 10 classes da base de dígitos MNIST.

manifolds) que nos dados originais estariam todas enoveladas.

Na seção seguinte utilizaremos o exemplo do reconhecimento de dígitos para ilustrar os conceitos introduzidos, indo de um modelo *shallow* para um modelo *deep*.

3.3. *From Shallow to Deep*

Vamos tomar como exemplo o problema de classificar imagens contendo dígitos numéricos dentre 10 dígitos (de 0 a 9). Para isso utilizaremos a base de dados MNIST que possui imagens de tamanho 28×28 (veja exemplos na Figura 3.1). Montando uma arquitetura de rede neural simples, poderíamos tentar resolver o problema da seguinte forma:

1. **entrada**: vetorizamos a imagem de entrada 28×28 de forma a obter um vetor $\mathbf{x}$ de tamanho 784×1;

2. **neurônios**: criamos 10 neurônios de saída, cada um representando a probabilidade da entrada pertencer a uma das 10 classes (dígitos 0,1,2,3,4,5,6,7,8 ou 9);

3. **pesos/parâmetros**: assim como em uma rede MLP, cada neurônio de saída está associado a pesos e termos bias que são aplicados nos dados para gerar uma combinação linear como saída;

4. **classificação**: para que a saída possa ser interpretada como probabilidades permitindo atribuir um exemplo à classe com maior probabilidade, utilizamos uma função conhecida como *softmax*.

Vamos recordar a montagem de uma rede neural e definir a notação. Na nossa arquitetura *shallow*, cada neurônio j processa uma imagem de entrada na forma $\mathbf{w}_j^t\mathbf{x}+\mathbf{b}_j$. Para ilustrar, se tivermos uma entrada com 4 valores no vetor, o neurônio j produziria o seguinte resultado:

$$\mathbf{w}_j^t\mathbf{x}+\mathbf{b}_j = (w_{j,1}x_1+b_1)+(w_{j,2}x_2+b_2)+(w_{j,3}x_3+b_3)+(w_{j,4}x_4+b_4)$$

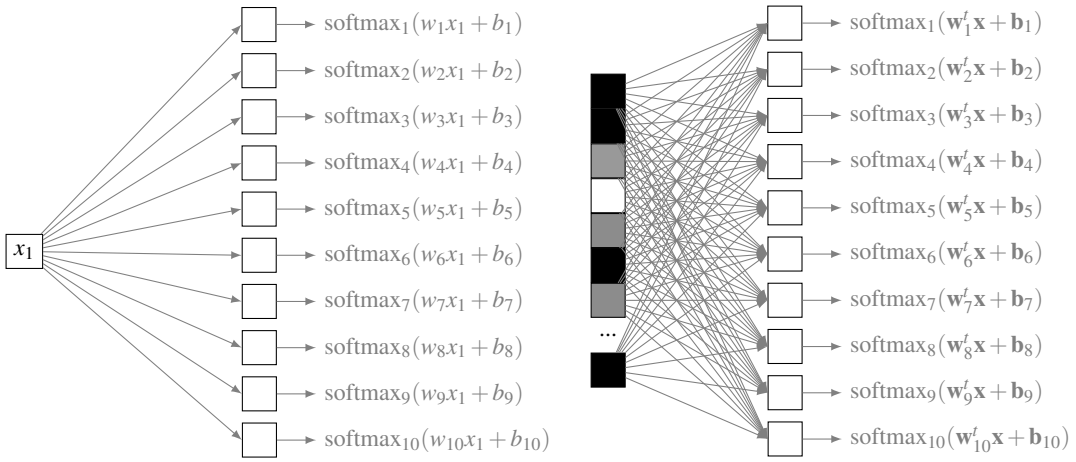
Como temos apenas uma camada precisamos que esse valor resultante do neurônio j seja o score da rede neural para a classe j. Assim, a partir do valor gerado pelo produto vetorial e somados os biases, aplicamos a função softmax, que é uma forma de obter valores normalizados para o intervalo 0-1 para cada classe c. A probabilidade de predizer

y como pertencendo à classe c, dada uma imagem de entrada $\mathbf{x}$, um conjunto de pesos, w, e ainda os termos *bias*, $\mathbf{b}$, ambos relativos ao neurônio da classe c é definida como:

$$P(y = c|\mathbf{x}; \mathbf{w}_c; \mathbf{b}_c) = \text{softmax}_c(\mathbf{x}^t \mathbf{w}_c + \mathbf{b}_c) = \frac{e^{\mathbf{x}^t \mathbf{w}_c + \mathbf{b}_c}}{\sum_j |e^{\mathbf{x}^t \mathbf{w}_j + \mathbf{b}_j}|}.$$

Note que então temos primeiro uma combinação linear $\mathbf{x}^t \mathbf{w}_c + \mathbf{b}_c$, depois exponenciamos esse valor, e dividimos pela soma da saída de todos os outros neurônios, de forma que a soma seja unitária, respeitando os critérios para uma distribuição de probabilidade. A função softmax é considerada uma **função de ativação** para classificação. Veremos posteriormente que existem diversas funções de ativação possíveis: cada qual adequada a um tipo de cenário.

Na Figura 3.2 mostramos a arquitetura pretendida. A título de ilustração mostramos dois casos, no primeiro teríamos como entrada apenas um escalar, enquanto que, no segundo caso, tem-se um vetor como entrada.



(a) um único valor x_1 como entrada (b) vetor $\mathbf{x}$ como entrada

Figura 3.2. Uma arquitetura *shallow*: (a) caso em que teríamos apenas um valor escalar de entrada e 10 neurônios de saída, (b) mostra um vetor de valores como entrada, assim cada um dos 10 neurônios de saída está associado a um vetor de pesos diferente.

Na prática, ao invés de utilizarmos uma única imagem de entrada por vez, utilizamos um conjunto de exemplos de entrada, chamado de **batch**. Para que isso possa ser feito, utilizaremos notação matricial. Agora, o resultado da combinação linear computada pelos neurônios será dado por:

$$\mathbf{X} \cdot \mathbf{W} + \mathbf{b} \tag{1}$$

$\mathbf{X}$ é uma matriz em que cada linha é uma imagem (ou seja, cada linha possui 784 valores, referentes aos 784 pixels da imagem). Aqui utilizaremos batches de tamanho 64, ou seja, a rede receberá como entrada 64 imagens de uma só vez e portanto a matriz $\mathbf{X}$ tem tamanho 64×784.

$\mathbf{W}$ é uma matriz que contém os pesos com os quais as imagens serão transforma-

das. Note que queremos produzir como saída, para cada imagem (linha da matriz **X**), as probabilidades para os 10 dígitos. Para isso a matriz **W** deverá ter 10 colunas e 784 linhas. Cada coluna contém os 784 pesos de um neurônio.

b é um vetor de termos bias: são utilizados os mesmos biases para todas as 64 imagens no batch. Assim a Equação 1 pode ser detalhada da seguinte forma:

$$\begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} & \cdots & x_{0,783} \\ x_{1,0} & x_{0,1} & x_{1,2} & \cdots & x_{0,783} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{63,0} & x_{63,1} & x_{63,2} & \cdots & x_{63,783} \end{bmatrix} \cdot \begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,9} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,9} \\ w_{2,0} & w_{2,1} & \cdots & w_{2,9} \\ \vdots & \vdots & \ddots & \vdots \\ w_{783,0} & w_{783,1} & \cdots & w_{783,9} \end{bmatrix} + \begin{bmatrix} b_0 \; b_1 \; b_2 \; ... \; b_9 \end{bmatrix}$$

Aplicamos então a função de ativação softmax para obter os resultados de classificação ou predições também em formato matricial **Y**. Note que essa será uma matriz com 64 linhas e 10 colunas, ou seja, para cada uma das 64 imagens no batch, temos as probabilidades para as 10 classes:

$$\mathbf{Y} = \text{softmax}(\mathbf{X} \cdot \mathbf{W} + \mathbf{b})$$

$$\mathbf{Y} = \begin{bmatrix} y_{0,0} & y_{0,1} & y_{0,2} & \cdots & y_{0,9} \\ y_{1,0} & y_{1,1} & y_{1,2} & \cdots & y_{1,9} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{63,0} & y_{63,1} & y_{63,2} & \cdots & y_{63,9} \end{bmatrix}$$

Agora seria importante questionar: quando **Y** terá boas predições para as imagens de entrada, ou seja, predizendo corretamente os dígitos? Isso acontecerá apenas se os parâmetros da rede (pesos **W** e bias **b**) forem adequados para o sistema. Mas como encontrar esses parâmetros e como saber quão bons esses parâmetros são?

Precisamos de uma função que compute a qualidade da predição realizada! Essa função é conhecida como **função de custo** (em inglês se utilizam os termos *loss function* ou *cost function*). Essa função é responsável por dizer quão longe estamos da predição ideal e portanto quantifica o "custo" ou "perda" ao aceitarmos a predição gerada pelos parâmetros atuais do modelo. Em outras palavras, qual é o custo de aceitarmos uma predição $\hat{y}$ sendo que a classe verdadeira é y? Para que possamos computar essa função precisamos nos basear em exemplos rotulados com suas classes verdadeiras, então o termo mais correto seria função de custo empírica.

Dentre as funções de custo mais utilizadas em classificação temos a **entropia cruzada** (*cross-entropy*). Considerando um único exemplo cuja distribuição de probabilidade de classes real é **y** e a predição é dada por $f(\mathbf{x}) = \hat{\mathbf{y}}$ temos a soma das entropias cruzadas (entre a predição e a classe real) de cada classe j:

$$\ell^{(\text{ce})} = -\sum_j y_j \cdot \log(\hat{y}_j + \varepsilon), \tag{2}$$

onde $\varepsilon << 1$ é uma variável para evitar $\log(0)$. Vamos assumir $\varepsilon = 0.0001$.

A título de exemplo sejam os seguintes vetores de distribuição de probabilidade de classes (note que ambas classe real e predita são a classe 7):

$$\mathbf{y} = \begin{bmatrix} 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \end{bmatrix}$$
$$\hat{\mathbf{y}} = \begin{bmatrix} 0.18 & 0.00 & 0.00 & 0.02 & 0.00 & 0.00 & 0.65 & 0.05 & 0.10 & 0.00 \end{bmatrix}$$

Então a entropia cruzada para esse exemplo seria:

$$\ell^{(\text{ce})} = -(0+0+0+0+0+0-0.6214+0+0+0) = 0.6214$$

Essa função recebe como entrada um vetor de scores, e produz valor 0 apenas no caso ideal em que $\mathbf{y} = \hat{\mathbf{y}}$. Em problemas de classificação, a entropia cruzada pode ser vista como a minimização da divergência de Kullback-Leibler entre duas distribuições de classes na qual a distribuição verdadeira tem entropia nula (já que possui um único valor não nulo, 1) [42]. Assim, temos a minimização da log-verossimilhança negativa da classe correta, o que relaciona essa equação também com a estimação de máxima verossimilhança.

Dado um conjunto atual de parâmetros $\mathbf{W}$ e $\mathbf{b}$, o custo completo com base em um batch (ou conjunto de treinamento) contendo N instâncias é geralmente computado por meio da média dos custos de todas as instâncias $\mathbf{x}_i$ e suas respectivas distribuições de classe: $\mathbf{y}_i$:

$$\mathscr{L}(\mathbf{W};\mathbf{b}) = \frac{1}{N}\sum_{i=1}^{N} \ell\left(\mathbf{y}_i, f(\mathbf{x}_i;\mathbf{W};\mathbf{b})\right).$$

Agora, precisamos ajustar os parâmetros de forma a minimizar $\mathscr{L}(\mathbf{W};\mathbf{b})$. Isso porque comumente esses parâmetros são inicializados com valores aleatórios, e queremos modificar esses valores iniciais de maneira a *convergir* para um modelo que nos dê boas predições.

Esse processo é feito utilizando algoritmos de otimização como o Gradiente Descendente (GD), que basicamente computa derivadas parciais de forma a encontrar, para cada parâmetro do modelo, qual modificação dos parâmetros permite minimizar a função. Veremos nas seções seguintes os métodos de otimização comumente utilizados em Deep Learning. Por enquanto assumiremos que o leitor está familiarizado com o Gradiente Descendente (GD) básico. Assim, executamos o treinamento por meio do algoritmo conhecido por Backpropagation, que irá atualizar os parâmetros de forma que a saída $\hat{\mathbf{y}}$ se aproxime do resultado esperado $\mathbf{y}$.

Treinamento nesse ponto, temos todos os componentes necessários para executar o algoritmo de treinamento. Inicializamos os parâmetros de forma aleatória, e então o algoritmo: (1) carrega 64 imagens sorteadas aleatoriamente do conjunto de treinamento, e (2) ajusta os parâmetros do modelo utilizando essas 64 imagens. Os passos (1) e (2) são repetidos por um número de vezes (iterações), até que o erro de classificação computado dentro do conjunto de treinamento seja suficientemente baixo, ou estabilize. Por exemplo, podemos definir 1000 iterações. Note que, nesse caso, o treinamento utilizará no máximo $64 \times 1000 = 64000$ exemplos (imagens) para aprender os parâmetros da rede.

Para mostrar como isso poderia ser feito na prática, abaixo mostramos um código na Listagem 3.1 em linguagem `Python` utilizando a biblioteca `Tensorflow` versão 1.2. Definimos primeiramente as variáveis: na linha 3 a matriz de imagens com $28 \times 28 \times 1$ (pois são em escala de cinza — para imagens RGB usar 3) além de um campo para indexar as imagens no batch (indicado por `None`); na linha 4 a matriz de pesos e na linha 5 o vetor de bias. O modelo considera as imagens redimensionadas para um vetor com 784 elementos (ver linha 10). A seguir a função de custo e método de otimização são definidos e 1000 iterações são executadas. Note que esse capítulo não pretende ser um tutorial sobre Tensorflow: o código é mostrado em alguns pontos para auxiliar no entendimento dos conceitos via implementação dos métodos.

Listagem 3.1. Treinamento de uma rede shallow com Tensorflow

```
1  import tensorflow as tf
2  # variaveis (matrizes e vetores)
3  X = tf.placeholder(tf.float32, [None, 28, 28, 1]) # batch de imagens X
4  W = tf.Variable(tf.zeros([784, 10])) # pesos
5  b = tf.Variable(tf.zeros([10])) # bias
6  Y = tf.placeholder(tf.float32, [None, 10]) # classes das imagens em X
7  inicia = tf.initialize_all_variables() # instancia inicializacao
8
9  # modelo que ira gerar as predicoes com base nas imagens vetorizadas
10 Y_ = tf.nn.softmax(tf.matmul(tf.reshape(X, [-1, 784]), W) + b)
11
12 # define funcao de custo (entropia cruzada)
13 entropia_cruzada = -tf.reduce_sum(Y * tf.log(Y_+0.0001))
14
15 # otimizacao com taxa de aprendizado 0.0025
16 otimiza = tf.train.GradientDescentOptimizer(0.0025)
17 treinamento = optimizer.minimize(entropia_cruzada)
18
19 sess = tf.Session() # instancia sessao
20 sess.run(inicia)    # executa sessao e inicializa
21
22 # executa 1000 iteracoes
23 for i in range(1000):
24     # carrega batch de 64 imagens (X) e suas classes (Y)
25     batch_X, batch_Y = mnist.train.next_batch(64)
26     dados_treinamento={X: batch_X, Y: batch_Y}
27
28     # treina com o batch atual
29     sess.run(treinamento, feed_dict=dados_treinamento)
30     # computa entropia-cruzada para acompanhar convergencia
31     ce = sess.run(entropia_cruzada, feed_dict=dados_treinamento)
```

3.3.1. Criando uma rede profunda

A rede anterior consegue alcançar uma acurácia próxima a 91% considerando o conjunto de testes da base de dados MNIST. Para que possamos melhorar esse resultado iremos utilizar uma arquitetura profunda, fazendo uso da composição de funções. Adicionaremos 2 camadas novas entre a entrada e a saída. Essas camadas são conhecidas como camadas ocultas (*hidden layers*). Em particular faremos uso de camadas completamente conectadas ou como se diz em terminologia Deep Learning, *fully connected* (FC), que é nada mais do que uma camada oculta assim como utilizada em redes MLP. Ilustramos a arquitetura na Figura 3.3. Teremos portanto uma predição alcançada por uma composição na forma:

$$\hat{\mathbf{y}} = f(\mathbf{x}) = f_3\left(f_2(f_1(\mathbf{x}_1;\mathbf{W}_1;\mathbf{b}_1);\mathbf{W}_2;\mathbf{b}_2)),\mathbf{W}_3;\mathbf{b}_3\right),$$

em que $f_1(\mathbf{x}_1) = \mathbf{x}_2$, $f_2(\mathbf{x}_2) = \mathbf{x}_3$ e finalmente $f_3(\mathbf{x}_3) = \mathbf{y}$.

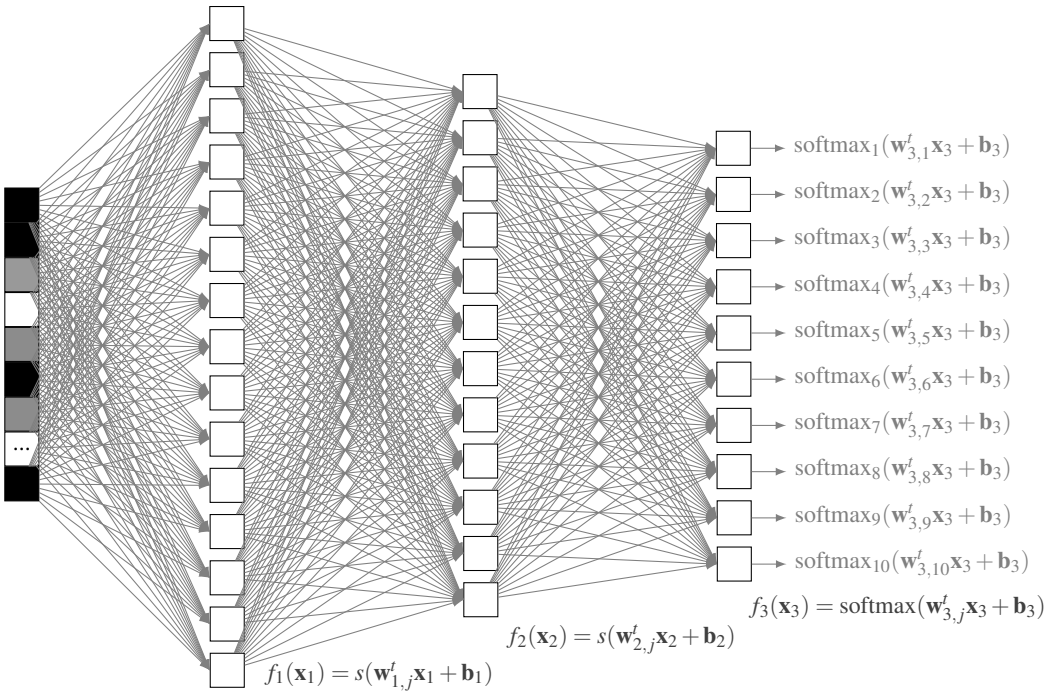
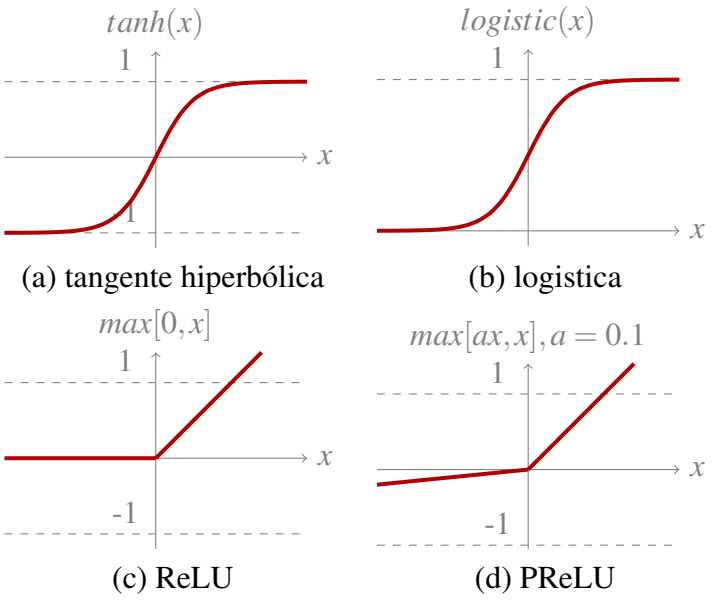


Figura 3.3. Uma arquitetura profunda com duas camadas ocultas, gerando representações intermediárias que antecedem a classificação. A quantidade de neurônios em cada camada é meramente ilustrativa.

Funções de ativação para as camadas FC ocultas a função de ativação softmax não é ideal! Em redes neurais é comum o uso de funções sigmoidais (como a logística e a tangente hiperbólica). Porém em Deep Learning, a função retificadora linear (*rectified linear function*, ReLU) tem sido mais utilizada por facilitar o processo de treinamento [36]. Isso porque as funções sigmoidais saturam a partir de um determinado ponto, enquanto a ReLU é simplesmente a função identidade para valores positivos. Veja a Figura 3.4 para exemplos de funções de ativação: as funções sigmoidais comprimem a saída para um intervalo curto, enquanto ReLU cancela todos os valores negativos, sendo linear para os positivos.

A função ReLU possui relações com a restrição de não-negatividade presente em regularização como em restauração de imagens utilizando projeções em subespaços [41]. Note ainda que ao calcularmos a derivada da ReLU, seu gradiente terá sempre uma direção não-nula, enquanto no caso das sigmoidais, para valores longe da origem podemos ter gradiente nulo. A ReLU paramétrica (PReLU) é uma variação que permite valores negativos com menor ponderação, parametrizado por uma variável $0 \leq a \leq 1$ [19]. Uma das possíveis vantagens da PReLU é a possibilidade de aprender a durante a fase de treinamento. No caso particular em que temos um valor fixo $a = 0.01$, temos a função conhecida por *Leaky ReLU*.

Na Listagem 3.2 mostramos como modificar o código anterior de forma a criar camadas intermediárias e utilizar a função ReLU para essas camadas.



(a) tangente hiperbólica (b) logistica

(c) ReLU (d) PReLU

Figura 3.4. Ilustração comparativa de funções de ativação: (c) ReLU é a mais utilizada em Deep Learning.

Listagem 3.2. Treinamento de uma rede deep com Tensorflow

```
# cria e inicializa aleatoriamente os pesos com distribuicao normal e sigma=0.1
W1 = tf.Variable(tf.truncated_normal([784, 64], stddev=0.1))
# bias sao inicializados com valores fixos 1/10
B1 = tf.Variable(tf.ones([64])/10)
W2 = tf.Variable(tf.truncated_normal([64, 32], stddev=0.1))
B2 = tf.Variable(tf.ones([32])/10)
W3 = tf.Variable(tf.truncated_normal([32, 10], stddev=0.1))
B3 = tf.Variable(tf.zeros([10]))

# entrada redimensionada
X1 = tf.reshape(X, [-1, 784])

# modelos das representacoes intermediarias
X2 = tf.nn.relu(tf.matmul(X1, W1) + B1)
X3 = tf.nn.relu(tf.matmul(X2, W2) + B2)

# saida da rede (a.k.a. logits)
X4 = tf.matmul(Y4, W5) + B5
# classificacao softmax
Y_ = tf.nn.softmax(X4)

# utilizamos uma funcao pronta no TF para calculo da entropia cruzada
entr_cruz = tf.nn.softmax_cross_entropy_with_logits(logits=X4, labels=Y_)
entr_cruz = tf.reduce_mean(entr_cruz)*100
```

Basicamente adicionamos novas matrizes de pesos e bias. Sendo que o números de neurônio de uma camada será o tamanho do seu vetor de saída. Assim, a matriz de pesos da camada seguinte deverá respeitar esse número. No código mostramos as camadas ocultas com 64 e 32 neurônios. Portanto $\mathbf{W}_1$ terá tamanho 784×64 (lembre que 784 é o tamanho do vetor de entrada), $\mathbf{W}_2$ terá tamanho 64×32 e finalmente $\mathbf{W}_3$ terá tamanho 32×10. Note também no código que a inicialização das matrizes de peso é feita usando números aleatórios obtidos de uma distribuição normal. Os bias são inicializados com valores pequenos, no exemplo todos iguais a 1/10.

Taxa de aprendizado com decaimento : é comum definir a taxa de aprendizado com decaimento (ao invés de fixa como no exemplo anterior). Isso significa que podemos começar com um valor mais alto, por exemplo algo entre 0.005 e 0.0025, e definir uma função de decaimento exponencial como $\exp^{-k/d}$, em que k é a iteração atual e d a taxa de decaimento (quanto maior, mais lento o decaimento).

Utilizando a base MNIST, com uma arquitetura parecida com a definida acima, é possível alcançar acurácias próximas a 97% no conjunto de testes. Isso foi alcançado por meio da inclusão de novas camadas, o que torna mais fácil encontrar os parâmetros corretos pois agora não é mais necessário aprender uma transformação direta de um vetor para uma classe, mas ao invés, aprendemos representações intermediárias, da mais simples, que processa o vetor, até a mais complexa, que prediz a classe da imagem. Sendo a MNIST uma base de imagens, para ir além do resultado atual precisamos utilizar um tipo de rede conhecida como Rede Convolucional.

3.4. Redes Convolucionais (CNNs)

Redes Neurais Convolucionais (CNNs) são provavelmente o modelo de rede Deep Learning mais conhecido e utilizado atualmente. O que caracteriza esse tipo de rede é ser composta basicamente de **camadas convolucionais**, que processa as entradas considerando campos receptivos locais. Adicionalmente inclui operações conhecidas como *pooling*, responsáveis por reduzir a dimensionalidade espacial das representações. Atualmente as CNNs de maior destaque incluem as Redes Residuais (ResNet) [18] e Inception [51].

A principal aplicação das CNNs é para o processamento de informações visuais, em particular imagens, pois a convolução permite filtrar as imagens considerando sua estrutura bidimensional (espacial). Considere o exemplo da base de dados MNIST apresentado na seção anterior. Note que ao vetorizar as imagens estamos desprezando toda a estrutura espacial que permite entender a relação entre os pixels vizinhos em uma determinada imagem. É esse tipo de estrutura que as CNNs tentam capturar por meio da camada convolucional.

3.4.1. Camada convolucional

Na camada convolucional cada neurônio é um filtro aplicado a uma imagem de entrada e cada filtro é uma matriz de pesos. Novamente, indicamos o livro texto [13] para uma introdução sobre convolução e filtros de imagens.

Seja uma imagem RGB de tamanho $224 \times 224 \times 3$ (o 3 indica os canais de cor R, G e B), que serve de entrada para uma camada convolucional. Cada filtro (neurônio) dessa camada irá processar a imagem e produzir uma transformação dessa imagem por meio de uma combinação linear dos pixels vizinhos. Note que agora não temos um peso para cada elemento da imagem. A título de exemplo em uma camada FC teríamos, para cada neurônio do nosso exemplo, 150528 pesos, um para cada valor de entrada. Ao invés, definimos filtros de tamanho $k \times k \times d$, em que k é a dimensão espacial do filtro (a ser definida) e d a dimensão de profundidade (essa depende da entrada da camada). Por exemplo, se definirmos $k = 5$ para a a primeira camada convolucional, então teremos filtros $5 \times 5 \times 3$, pois como a imagem possui 3 canais (RGB), então $d = 3$, e assim cada neurônio terá $5 \times 5 \times 3 = 75$ pesos.

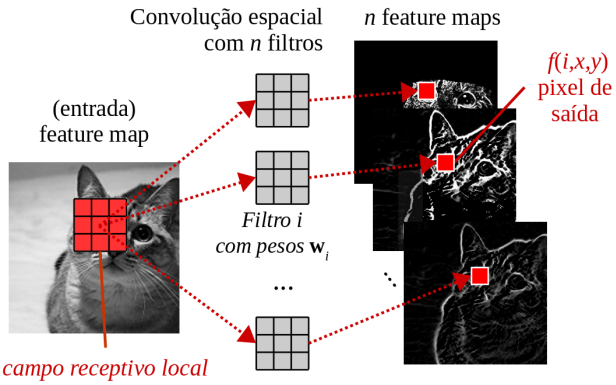


Figura 3.5. Ao utilizar convolução, processa-se informações locais utilizando cada posição (x,y) como centro: essa região é chamada de campo receptivo. Seus valores são então usados como entrada para um filtro i com parâmetros $\mathbf{w}_i$, produzindo um único valor (pixel) no mapa de características $f(i,x,y)$ gerado como saída.

Cada região da imagem processada pelo filtro é chamada de campo receptivo local (*local receptive field*); um valor de saída (pixel) é uma combinação dos pixels de entrada nesse campo receptivo local (veja Figura 3.5). No entanto todos os campos receptivos são filtrados com os mesmos pesos locais para todo pixel. Isso é o que torna a camada convolucional diferente da camada FC. Assim, um valor de saída ainda terá o formato bidimensional. Por exemplo podemos dizer que $f_{l+1}(i,x,y)$ é o pixel resultante da filtragem da imagem vinda da camada anterior l, processada pelo filtro i a partir dos valores da vizinhança centrados na posição (x,y). No nosso exemplo com $k = 5$, teremos uma combinação linear de 25 pixels da vizinhança para gerar um único pixel de saída.

Os tamanhos de filtros mais utilizados são $5 \times 5 \times d$, $3 \times 3 \times d$ e $1 \times 1 \times d$. Como trabalhamos com matrizes multidimensionais (com profundidade d), utilizamos o termo **tensor** para denotá-las.

Considere um problema em que temos como entrada imagens RGB, de tamanho $64 \times 64 \times 3$. Sejam então duas camadas convolucionais, a primeira com 4 filtros de tamanho $k_1 = 5$, e a segunda com 5 filtros de tamanho $k_2 = 3$. Considere ainda que a convolução é feita utilizando extensão da imagem com preenchimento por zeros (*zero padding*) de forma que conseguimos realizar a filtragem para todos os pixels da imagem, mantendo seu tamanho. Nesse cenário teríamos a seguinte composição:

$$\hat{\mathbf{y}} = f(\mathbf{x}) = f_2(f_1(\mathbf{x}_1;\mathbf{W}_1;\mathbf{b}_1);\mathbf{W}_2;\mathbf{b}_2)),$$

em que $\mathbf{W}_1$ possui dimensão $4 \times 5 \times 5 \times 3$ (4 filtros de tamanho 5×5, entrada com profundidade 3), e portanto a saída da camada 1, $\mathbf{x}_2 = f_1(\mathbf{x}_1)$ terá tamanho: $64 \times 64 \times 4$. Após a convolução utiliza-se uma função de ativação (comumente a função ReLU já descrita anteriormente) que trunca para zero os pixels negativos. A Figura 3.6 ilustra esse processo, bem como o da camada 2, que recebe por entrada o tensor $64 \times 64 \times 4$. Essa segunda camada possui 5 filtros $3 \times 3 \times 4$ (já que a profundidade do tensor de entrada tem $d = 4$), e gera como saída $\mathbf{x}_3 = f_2(\mathbf{x}_2)$, um tensor de tamanho $64 \times 64 \times 5$.

Chamamos de mapa de características a saída de cada neurônio da camada convo-

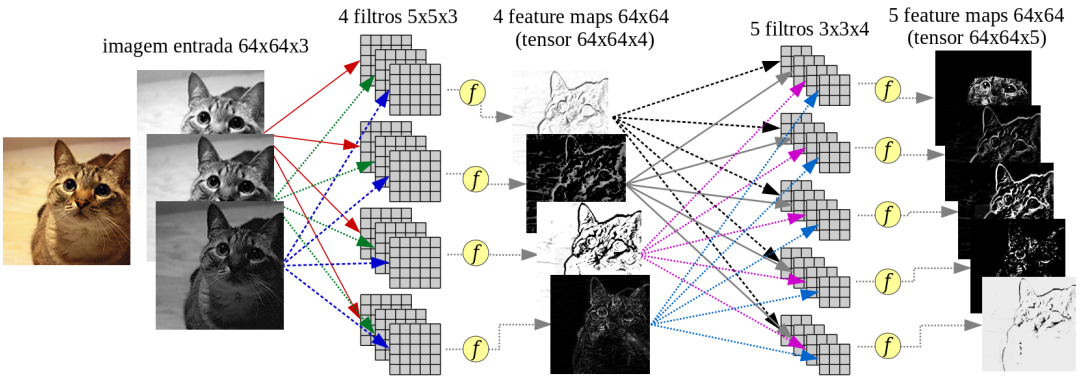


Figura 3.6. Ilustração de duas camadas convolucionais: a primeira com 4 filtros $5 \times 5 \times 3$, que recebe como entrada uma imagem RGB $64 \times 64 \times 3$, e produz um tensor com 4 feature maps; a segunda camada convolucional contém 5 filtros $3 \times 3 \times 4$ que filtram o tensor da camada anterior, produzindo um novo tensor de feature maps com tamanho $64 \times 64 \times 5$. Os círculos após cada filtro denotam funções de ativação, como por exemplo a ReLU.

lucional (mais detalhes na seção seguinte). Antes de prosseguir, outro aspecto importante para se mencionar é o passo ou *stride*. A convolução convencionais é feita com passo/stride 1, ou seja, filtramos todos os pixels e portanto para uma imagem de entrada de tamanho 64×64, geramos uma nova imagem de tamanho 64×64. O uso de strides maiores que 1 é comum quando deseja-se reduzir o tempo de execução, pulando pixels e assim gerando imagens menores. Ex. com stride = 2 teremos como saída uma imagem de tamanho 32×32.

3.4.2. Feature maps (mapas de características)

Cada representação gerada por um filtro da camada convolucional é conhecida como "mapa de características", do inglês *feature map* ou *activation map*. Utilizaremos *feature map* pois é o termo mais comum na literatura. Os mapas gerados pelos diversos filtros da camada convolucional são empilhados, formando um tensor cuja profundidade é igual ao número de filtros. Esse tensor será oferecido como entrada para a próxima camada como mostrado na Figura 3.6. Note que, como a primeira camada convolucional gera um tensor $64 \times 64 \times 4$, os filtros da segunda camada terão que ter profundidade 4. Caso adicionássemos uma terceira camada convolucional, os filtros teriam que ter profundidade 5.

3.4.3. Pooling

É comum reduzir a dimensão espacial dos mapas ao longo das camadas da rede. Essa redução em tamanho é chamada de *pooling* sendo a operação de máximo *maxpooling* comumente empregada. Essa operação tem dois propósitos: em primeiro lugar, o custo computacional, pois como a profundidade dos tensores, d, costuma aumentar ao longo das camadas, é conveniente reduzir a dimensão espacial dos mesmos. Em segundo, reduzindo o tamanho das imagens obtemos um tipo de composição de banco de filtros multi-resolução que processa imagens em diferentes espaços-escala. Há estudos a favor de não utilizar pooling, mas aumentar o stride nas convoluções [49], o que produz o mesmo efeito redutor.

3.4.4. Camadas fully connected (FC)

Já mencionamos o uso de camadas FC, que são camadas presentes em redes neurais MLP. Nesse tipo de camada cada neurônio possui um peso associado a cada elemento do vetor de entrada. Em CNNs utiliza-se camadas FC posicionadas após múltiplas camadas convolucionais. A transição entre uma camada convolucional (que produz um tensor) e uma camada FC, exige que o tensor seja vetorizado. Por exemplo, se a camada convolucional antes de uma camada FC gera um tensor $4 \times 4 \times 40$, redimensionamos esses dados de forma que ele possua tamanho $1 \times (4 \times 4 \times 40) = 1 \times 640$. Assim, cada neurônio na camada FC deverá possuir 640 pesos de forma a produzir uma combinação linear do vetor.

Como descrito anteriormente, arquiteturas mais recentes utilizam camadas FC ocultas com função de ativação ReLU, e a camada de saída (classificador) com função de ativação softmax.

3.4.5. Arquiteturas de CNNs e seus parâmetros

CNNs tradicionais são combinação de blocos de camadas convolucionais (Conv) seguidas por funções de ativação, eventualmente utilizando também *pooling* (Pool) e então uma séria de camadas completamente conectadas (FC), também acompanhadas por funções de ativação, da seguinte forma:

$$\text{CNN} \equiv P \times [C \times (\text{Conv} \rightarrow \text{AF}) \rightarrow \text{Pool}] \rightarrow F \times [\text{FC} \rightarrow \text{AF}]$$

Para criar uma CNN, deve-se definir os seguintes hiper-parâmetros: número de camadas convolucionais C (para cada camada, o número de filtros, seu tamanho e o tamanho do passo dado durante a convolução), número de camadas de `pooling` P (sendo nesse caso necessário escolher também o tamanho da janela e do passo que definirão o fator de subamostragem), o número de camadas totalmente conectadas F (e o número de neurônios contidos em cada uma dessas camadas).

Note no entanto que algumas redes mais recentes também tem empregado a pré-ativação (a função AF é aplicada nos dados de entrada, antes da convolução ou camada FC).

O número de parâmetros em uma CNN está relacionado basicamente, aos valores a serem aprendidos em todos todos os filtros nas camadas convolucionais, os pesos das camadas totalmente conectadas e os bias.

—*Exemplo*: considere uma arquitetura para analisar imagens RGB com dimensão $64 \times 64 \times 3$ cujo objetivo é classificar essas imagens em 5 classes. Essa arquitetura será composta por três camadas convolucionais, duas *max pooling*, e duas camadas totalmente conectadas, da seguinte forma:

- Conv.L 1: 10 filtros $5 \times 5 \times 3$, saída: tensor de dimensão $64 \times 64 \times 10$
- *Max pooling* 1: subamostragem com fator 4 (janela de tamanho 2×2 e stride 2), saída: tensor de dimensão $16 \times 16 \times 10$
- Conv.L2: 20 filtros $3 \times 3 \times 10$, saída: tensor de dimensão $16 \times 16 \times 20$

- Conv.L3: 40 filtros $1 \times 1 \times 20$, saída: tensor de dimensão $16 \times 16 \times 40$
- *Max pooling* 2: subamostragem com fator 4 (janela de tamanho 2×2 e stride 2), saída: tensor de dimensão $4 \times 4 \times 40$
- FC.L1: 32 neurônios, saída: 32 valores
- FC.L2 (saída da rede): 5 neurônios (um por classe), saída: 5 valores

Considerando que cada um dos filtros das três camadas convolucionais tem $p \times q \times d$ parâmetros, além do bias, e que as camadas totalmente conectadas possuem pesos e o termo bias associado com cada valor do vetor recebido da camada anterior, então o número total de parâmetros nessa arquitetura é:

$$\begin{aligned} & (10 \times [5 \times 5 \times 3 + 1] = 760) && \text{[Conv.L1]} \\ + & (20 \times [3 \times 3 \times 10 + 1] = 1820) && \text{[Conv.L2]} \\ + & (40 \times [1 \times 1 \times 20 + 1] = 840) && \text{[Conv.L3]} \\ + & (32 \times [640 + 1] = 20512) && \text{[FC.L1]} \\ + & (5 \times [32 + 1] = 165) && \text{[FC.L2]} \\ = & \ 24097 \end{aligned}$$

É possível perceber que, apesar de ser uma arquitetura relativamente pequena, o número de parâmetros as serem aprendidos pode ser grande e cresce consideravelmente quanto maior a arquitetura utilizada.

3.4.6. Implementação de CNN para o caso MNIST

Voltando ao exemplo da classificação de dígitos usando a base MNIST, poderíamos projetar camadas convolucionais de forma a extrair representações com base na estrutura espacial das imagens. Para isso, mostramos na Listagem 3.3 linhas de código para criar uma camada convolucional: na linha 2 criamos a matriz $\mathbf{W}_1$ utilizando 4 filtros de tamanho $5 \times 5 \times 3$, definimos stride 1 na linha 5 e a convolução 2d na linha 6, com uso da função ReLU como ativação na linha 7, produzindo o tensor $\mathbf{x}_2$.

Listagem 3.3. Implementação de camadas convolucionais

```
1 # inicializacao dos parametros W e B da camada convolucional 1
2 W1 = tf.Variable(tf.truncated_normal([5, 5, 3, 4], stddev=0.1))
3 B1 = tf.Variable(tf.ones([4])/10) # 2 e' o numero de mapas de saida da camada
4
5 stride = 1  # mantem a imagem no tamanho original
6 Xconv1 = tf.nn.conv2d(X1, W1, strides=[1, stride, stride, 1], padding='SAME')
7 X2 = tf.nn.relu(Xconv1 + B1)
```

Utilizando uma CNN com 3 camadas convolucionais com a seguinte configuração, na sequência, conv1: 8 filtros 5×5 stride 1, conv2: 16 filtros 3×3 stride 2, conv3: 32 filtros 3×3 stride 2, FC1 oculta com 256 neurônios, FC2 de saída com 10 neurônios, é possível alcançar acurácias próximas a 99%. No entanto, sempre que ajustamos demais o modelo nos dados de treinamento — e isso ocorre particularmente em Deep Learning pois o número de parâmetros é muito grande — há um sério risco do processo

de treinamento resultar na memorização dos exemplos que foram utilizados nessa etapa, gerando um efeito conhecido por *overfitting*. Iremos discutir esse ponto na Seção 3.5.1 como umas das limitações dos métodos baseados em DL. Por enquanto veremos como realizar o treinamento da melhor forma possível, buscando evitar esse efeito e atingir uma convergência aceitável do modelo de forma a ser útil nas tarefas que dispomos.

3.4.7. Algoritmos de otimização

Com uma função de custo definida, precisa-se então ajustar os parâmetros de forma que o custo seja reduzido. Para isso, em geral, usa-se o algoritmo do Gradiente Descendente em combinação com o método de backpropagation, que permite obter o gradiente para a sequência de parâmetros presentes na rede usando a regra da cadeia. Existem diversos materiais disponíveis que apresentam explicações detalhadas sobre o Gradiente Descendente e como funciona o backpropagation. Assim, assume-se que o leitor esteja familiarizado com conceitos fundamentais de ambos os algoritmos, focando em seu funcionamento para CNNs.

Como o cálculo de $\mathscr{L}(W)$ é feito baseado em uma amostra finita da base de dados, calcula-se estimativas de Montecarlo da distribuição real que gera os parâmetros. Além disso, é importante lembrar que CNNs possuem muitos parâmetros que precisam ser aprendidos fazendo com que seja necessário treiná-la usando milhares, ou até milhões, de imagens (diversas bases de dados atualmente possuem mais de 1TB de dados). Entretanto, realizar a otimização usando milhões de instâncias torna a utilização de Gradiente Descendente inviável, uma vez que esse algoritmo calcula o gradiente para todas as instâncias individualmente. Essa dificuldade fica clara quando se pensa que para executar uma época (i.e., executar o algoritmo para todas as instâncias na base de treinamento) seria necessário carregar todas as instâncias para uma memória limitada, o que não seria possível. Algumas alternativas foram propostas para superar esse problema. Algumas dessas alternativas são descritas abaixo: SGD, Momentum, AdaGrad, RMSProp e Adam.

Gradiente Descendente Estocástico (do inglês Stochastic Gradient Descent, SGD) uma forma de acelerar o treinamento é utilizando métodos que oferecem aproximações do Grandiente Descendente, por exemplo usando amostras aleatórias dos dados ao invés de analisando todas as instâncias existentes. Por esse motivo, o nome desse método é Gradiente Descendente Estocástico, já que ao invés de analisar todos dados disponíveis, analisa-se apenas uma amostra, e dessa forma, adicionando aleatoriedade ao processo. Também é possível calcular o Gradiente Descende usando apenas uma instância por vez (método mais utilizado para analisar fluxos de dados ou aprendizagem online). Na prática, o mais comum é utilizar os chamados *mini-batches* (amostra aleatória dos dados) com um tamanho fixo B. Após executar diversas iterações (sendo que cada iteração irá adaptar os parâmetros usando as instâncias no mini-batch atual), espera-se obter uma aproximação do método do Gradiente Descendente.

$$W_{t+1} = W_t - \eta \sum_{j=1}^{B} \nabla \mathscr{L}(W; x_j^{\mathrm{B}}),$$

na qual η é o parâmetro que define a taxa de aprendizado (em inglês *learning rate*), ou seja, esse parâmetro é utilizado para definir o tamanho do passo que será dado na direção apontada pelo gradiente. É comum utilizar valores altos para η no começo do treinamento e fazer com ele decaia exponencialmente em função do número de iterações executadas.

Devido ao seu fator aleatório, SGD fornece uma aproximação grosseira do Gradiente Descendente, assim, fazendo com que a convergência não seja suave. Por esse motivo, outras variantes foram propostas buscando compensar esse fato, tais como AdaGrad (*Adaptive Gradient*) [9], AdaDelta (*Adaptive learning rate*) [56] and Adam (*Adaptive moment estimation*) [24]. Tais variantes baseiam-se nos conceitos de momentum e normalização, que serão descritos abaixo.

Momentum adiciona um novo hiper-parâmetro que permite controlar a velocidade das mudanças nos parâmetros W da rede. Isso é feito criando um fator de *momentum*, que dá peso para a atual direção do gradiente, e previne que uma nova atualização dos parâmetros W_{t+1} se desvie muito da atual direção de busca no espaço dos parâmetros:

$$W_{t+1} = W_t + \alpha(W_t - W_{t-1}) + (1-\alpha)\left[-\eta\nabla\mathscr{L}(W_t)\right],$$

onde $\mathscr{L}(W_t)$ é a perda calculada usando algumas instâncias (usualmente um mini-batch) e os parâmetros atuais W_t. É importante notar como o tamanho do passo na iteração $t+1$ agora é limitado pelo passo dado na iteração anterior, t.

AdaGrad busca dar mais importância a parâmetros pouco utilizados. Isso é feito mantendo um histórico de quanto cada parâmetro influenciou o custo, acumulando os gradientes de forma individual $g_{t+1} = g_t + \nabla\mathscr{L}(W_t)^2$. Essa informação é então utilizada para normalizar o passo dado em cada parâmetro:

$$W_{t+1} = W_t - \frac{\eta\nabla\mathscr{L}(W_t)^2}{\sqrt{g_{t+1}} + \varepsilon},$$

como o gradiente é calculado com base no histórico e para cada parâmetro de forma individual, parâmetros pouco utilizados terão maior influência no próximo passo a ser dado.

RMSProp calcula médias da magnitude dos gradientes mais recentes para cada parâmetro e as usa para modificar a taxa de aprendizado individualmente antes de aplicar os gradientes. Esse método é similar ao AdaGrad, porém, nele g_t é calculado usando uma média com decaimento exponencial e não com a simples soma dos gradientes:

$$g_{t+1} = \gamma g_t + (1-\gamma)\nabla\mathscr{L}(W_t)^2$$

g é chamado o momento de segunda ordem de $\nabla\mathscr{L}$ (o termo momento tem a ver com o gradiente não confundir com o efeito *momentum*). A atualização dos parâmetros é então

feita adicionando-se um tipo de momentum:

$$W_{t+1} = W_t + \alpha(W_t - W_{t-1}) + (1-\alpha)\left[-\frac{\eta \nabla \mathscr{L}(W_t)}{\sqrt{g_{t+1}} + \varepsilon}\right],$$

Adam utiliza uma idéia similar ao AdaGrad e ao RMSProp, contudo o *momentum* é usado para tanto para o momento (novamente, não confundir com *momentum*) de primeira quanto o de segunda ordens, tendo assim α e γ para controlar W e g, respectivamente. A influência de ambos diminui com o tempo de forma que o tamanho do passo diminua conforme aproxima-se do mínimo. Uma variável auxiliar m é usada para facilitar a compreensão:

$$m_{t+1} = \alpha_{t+1} g_t + (1-\alpha_{t+1}) \nabla \mathscr{L}(W_t)$$
$$\hat{m}_{t+1} = \frac{m_{t+1}}{1-\alpha_{t+1}}$$

m é o momento de primeira ordem de $\nabla \mathscr{L}$ e $\hat{m}$ é m após a aplicação do fator de decaimento. Então, precisa-se calcular os gradientes g usado para normalização:

$$g_{t+1} = \gamma_{t+1} g_t + (1-\gamma_{t+1}) \nabla \mathscr{L}(W_t)^2$$
$$\hat{g}_{t+1} = \frac{g_{t+1}}{1-\gamma_{t+1}}$$

g é o momento de segunda ordem de $\nabla \mathscr{L}$. A atualização dos parâmetros é então calculada da seguinte forma:

$$W_{t+1} = W_t - \frac{\eta \hat{m}_{t+1}}{\sqrt{\hat{g}_{t+1}} + \varepsilon}$$

3.4.8. Aspectos importantes no treinamento de CNNs

Inicialização a inicialização dos parâmetros é importante para permitir a convergência da rede. Atualmente, se utiliza números aleatórios sorteados a partir de uma distribuição Gaussiana $\mathscr{N}(\mu, \sigma)$ para inicializar os pesos. Pode-se utilizar um valor fixo como o $\sigma = 0.01$ utilizado nos nossos exemplos anteriores. Porém o uso desse valor fixo pode atrapalhar a convergência [26]. Como alternativa, recomenda-se usar $\mu = 0$, $\sigma = \sqrt{2/n_l}$, onde n_l é o número de conexões na camada l, e inicializar vetores bias com valores constantes, conforme fizemos nos nossos exemplos ou ainda iguais a zero [19].

Tamanho do Minibatch devido ao uso de SGD e suas variantes, é preciso definir o tamanho do minibatch de exemplos a ser utilizado em cada iteração no treinamento. Essa escolha deve levar em consideração restrições de memória mas também os algoritmos de otimização empregados. Um tamanho de batch pequeno pode tornar mais difícil a minimização do custo, mas um tamanho muito grande também pode degradar a velocidade de convergência do SGD para funções objetivo convexas [30].

Apesar disso, foi demonstrado que até certo ponto um valor maior para o tama-

nho do batch B ajuda a reduzir a variância das atualizações em cada iteração do SGD (pois essa usa a média do custo dos exemplos do batch), permitindo que possamos utilizar tamanhos de passo maiores [3]. Ainda, minibatches maiores são interessantes para o caso de treinamento usando GPUs, gerando maior throughput pelo uso do algoritmo de backpropagation com reuso dos dados por meio da multiplicação entre matrizes (em contrapartida ao uso de várias multiplicações vetor-matriz). Via de regra, escolhe-se B de forma a ocupar toda a memória disponível na GPU, e a maior taxa de aprendizado possível.

Olhando as arquiteturas mais populares (VGGNet [48], ResNet [18], Inception [51, 50]), notamos o uso desde 32 até 256 exemplos por batch. Num caso mais extremo, um artigo recente mostro o uso de $B = 8192$, o que foi alcançado com uso de 256 GPUs em paralelo e uma regra de ajuste de escala para a taxa de aprendizado. Com isso os autores conseguiram treinar uma ResNet na base ImageNet em apenas 1 hora [16].

Regularização quando usamos uma função de custo para realizar o treinamento de uma rede neural, como apresentado anteriormente, alguns problemas podem surgir. Um caso típico é existência de diversos parâmetros W que façam com que o modelo classifique corretamente o conjunto de treinamento. Como há múltiplas soluções, isso torna mais difícil encontrar bons parâmetros.

Um termo de regularização adicionado à função de custo auxilia a penalizar essa situação indesejada. A regularização mais comumente utilizada é a de norma L2, que é a soma dos quadrados dos pesos. Como queremos minimizar também essa norma em adição à função de custo, isso desencoraja a existência de alguns poucos valores altos nos parâmetros, fazendo com que as funções aprendidas aproveitem todo o espaço de parâmetros na busca pela solução. Escrevemos a regularização na forma:

$$\mathscr{L}(W) = \frac{1}{N}\sum_{j=1}^{N} \ell\left(y_j, f(x_j;W)\right) + \lambda R(W).$$

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

onde λ é um hiper-parâmetro utilizado para ponderar a regularização, ditando o quanto permitimos que os parâmetros em W possam crescer. Valores para λ podem ser encontrados realizando testes por validação cruzada no conjunto de treinamento.

Dropout é uma técnica para minimizar overfitting proposta em [20] que, na fase de treinamento e durante o passo de propagação dos dados pela rede, aleatoriamente desativa com probabilidade p a ativação de neurônios (em particular neurônios de camadas FC).

Esse procedimento pode ser considerado uma forma de regularização pois, ao desativar aleatoriamente neurônios, perturba os feature maps gerados a cada iteração por meio da redução da complexidade das funções (já que a composição utiliza menos ativações). Ao longo das iterações isso dá um efeito que minimiza a variância e aumenta o

viés das funções, e por isso foi demonstrado que o *dropout* tem relações com o método de ensemble Bagging [54]. A cada iteração do SGD, cria-se uma rede diferente por meio da subamostragem das ativações. Na fase de teste, no entanto, não se utiliza dropout, e as ativações são re-escaladas com fator p para compensar as ativações que foram desligadas durante a fase de treinamento.

Batch normalization (BN) as CNNs do estado da arte (em particular Inception [51, 50] e ResNet [18]) utilizam a normalização de batches (BN, Batch normalization) [23].

Como alternativa, na saída de cada camada pode-se normalizar o vetor de formas diferentes. O método canal-por-canal normaliza os mapas, considerando cada feature map individualmente ou considerando também mapas vizinhos. Podem ser empregadas normalizações L1, L2 ou variações. Porém esses métodos foram abandonados em favor do BN.

BN também tem efeito regularizador, normalizando as ativações da camada anterior em cada batch de entrada, mantendo a ativação média próxima a 0 (centralizada) e o desvio padrão das ativações próximo a 1, e utilizando parâmetros γ e β para compor a transformação linear do vetor normalizado:

$$\mathrm{BN}_{\gamma,\beta}(x_i) = \gamma\left(\frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}\right) + \beta. \tag{3}$$

Note que γ e β podem ser incluídos no modelo de forma a serem ajustados/aprendidos durante o backpropagation [23], o que pode ajustas a normalização a partir dos dados, e até mesmo cancelar essa normalização, i.e. se esses forem ajustados para $\gamma = \sqrt{\sigma_B^2}$ e $\beta = \mu_B$.

BN se tornou um método padrão nos últimos anos, tendo substituído o uso de regularização e em alguns casos até mesmo tornando desnecessário o uso de dropout.

Data-augmentation como mencionado anteriormente, redes profundas possuem um espaço de parâmetros muito grande a ser otimizado. Isso faz com que seja necessário dispor de um número muitas vezes proibitivo de exemplos rotulados. Assim, podem ser empregados métodos de *data augmentation*, i.e. geração de uma base de treinamento aumentada. Isso é feito comumente aplicando operações de processamento de imagens em cada imagem da base gerando 5, 10 (o um mais) novas imagens por imagem original existente.

Alguns métodos para geração de imagens incluem [5]: (i) corte da imagem de entrada em diferentes posições — tirando vantagem do fato de que as imagens possuem resolução superior à entrada comumente esperada pelas CNNs (e.g. 224×224), ao invés de redimensionarmos uma imagem para esse valor, cortamos a imagem original de maior resolução em diversas subimagens de tamanho 224×224; (ii) girar a imagem horizontalmente, e também verticalmente caso faça sentido no contexto das imagens de entrada, como por exemplo é o caso de imagens astronômicas, de sensoriamento remoto, etc.; (iii)

adicionando ruído às imagens de entrada [37]; (iv) criando imagens por meio da aplicação de Análise de Componentes Principais (PCA) nos canais de cor, como no método Fancy PCA [26].

Pre-processamento é possível pré-processar as imagens de entrada de diversas formas. As mais comuns incluem: (i) computar a imagem média para todo o conjunto de treinamento e subtrair essa média de cada imagem; (ii) normalização z-score, (iii) PCA *whitening* que primeiramente tenta descorrelacionar os dados projetando os dados originais centrados na origem em uma auto-base (em termos dos auto-vetores), e então dividindo os dados nessa nova base pelos auto-valores relacionados de maneira a normalizar a escala.

O método z-score é o mais utilizado dentre os citados (centralização por meio da subtração da média, e normalização por meio da divisão pelo desvio padrão), o que pode ser visto como uma forma de realizar *whitening* [29].

3.4.9. Utilizando modelos pré-treinados: fine-tuning e extração de características

Comumente dispomos de um conjunto de dados pequeno, inviáveis para permitir o treinamento de uma CNN a partir do zero, mesmo empregando métodos de data augmentation (note que a geração de imagens apenas cria versões perturbadas das mesmas imagens). Nesse caso é muito útil utilizar um modelo cujos parâmetros já foram encontrados para um conjunto de dados grande (como por exemplo para a base ImageNet [8] que possui mais de um milhão de imagens de 1000 classes).

O processo de **fine-tuning** consiste basicamente de continuar o treinamento a partir dos pesos iniciais, mas agora utilizando um subconjunto de sua base de dados. Note que é provável que essa nova base de dados tenha outras classes (em contrapartida por exemplo às 1000 classes da ImageNet). Assim se você deseja criar um novo classificador será preciso remover a última camada e adicionar uma nova camada de saída com o número de classes desejado, a qual deverá ser treinada do zero a partir de uma inicialização aleatória.

A partir disso, há várias abordagens para realizar fine-tuning, que incluem por exemplo (i) permitir que o algoritmo ajuste todos os pesos da rede com base nas novas imagens, (ii) congelar algumas camadas e permitir que o algoritmo ajuste apenas os parâmetros um subconjunto de camadas – por exemplo podemos ajustar apenas os pesos da última camada criada, ou apenas os pesos das FCs, etc, (iii) criar novas camadas adicionais com números e tamanhos de filtros diferentes.

O mais comum é a abordagem (ii), congelando das primeiras camadas (em geral convolucionais) e permitindo que as camadas mais profundas se adaptem.

Para obter **extração de características**, mesmo sem realizar fine-tuning, oferece-se como entrada as imagens desejadas, e utilizamos como vetor de características a saída de uma das camadas da rede (antes da camada de saída). Comumente se utiliza a penúltima camada: por exemplo na VGGNet [48] a FC2 tem 4096 neurônios, gerando um vetor de 4096 elementos, já na Inception V3 [51] temos 2048 elementos na penúltima camada. Caso a dimensionalidade seja alta, é possível utilizar algum método de redução de dimensionalidade ou quantização baseada por exemplo em PCA [43] ou Product

Quantization [4].

3.5. Porque Deep Learning funciona?

Ao longo desse capítulo esperamos que o leitor tenha compreendido com clareza o funcionamento dos métodos de DL, incluindo os componentes básicos das redes profundas (tipos de camadas, representações, funções de ativação, funções de custo etc.) e os algoritmos e técnicas fundamentais (otimização, regularização, normalização, etc.). Mostramos que, combinando uma arquitetura adequada em termos do número de camadas, quantidade de neurônios por camada, e atentando para detalhes no processo de otimização, é possível alcançar excelentes resultados na regressão e classificação de sinais de alta dimensionalidade. No entanto resta a dúvida: porque esses métodos funcionam tão bem? É justificada a grande atenção atualmente dispensada a esses métodos, e a expectativa que os cerca em termos da solução de outros problemas? Quais as garantias teóricas que embasam seu funcionamento?

Nas seções anteriores mostramos que a hipótese para o sucesso dos métodos de DL está na composição de funções, que realiza transformações sucessivas a partir do vetor de entrada. No entanto, analisar esses algoritmos é difícil devido à falta de estrutura para compreensão das suas invariantes, bem como das transformações intrínsecas.

O papel da profundidade : há alguns estudos que se dedicam a demonstrar porque redes profundas são melhores do que redes superficiais a partir de teoremas de aproximação. Um dos trabalhos nessa direção é o de Kolmogorov (1936) [25] que demonstrou um limite superior para a aproximação de funções contínuas por meio de subespaços de funções, seguido por Warren (1968) que encontrou limites para a aproximação por meio de polinômios, em termos de graus polinomiais e dimensão da função alvo, provando a dimensão VC de polinômios [55]. O refinamento desses resultados veio com teoremas de hierarquia de profundidade em complexidade de circuitos [22], demonstrando a diferença entre circuitos com uma certa profundidade de circuitos de menor profundidade. Esse resultado é uma das bases para o estudo de redes neurais profundas [35] [52] [31].

Entre os resultados obtidos [31] demonstra que o custo de achatarmos o grafo formado pela rede neural para um determinado problema é comumente proibitivo, relacionando o uso de múltiplas camadas com algoritmos de divisão e conquista. Um exemplo apontado pelos autores é a Transformada Rápida de Fourier (FFT), que basicamente faz uso de uma fatorização esparsa da matriz da Transformada Discreta de Fourier (DFT), resultando em $O(n \log n)$ elementos não-nulos na matrix, em contraste com os n^2 elementos não nulos na matriz da DFT. Assim, se imaginarmos a FFT como operações organizadas em um grafo (rede), achatar as operações em uma única camada irá aumentar a contagem de $n \log n$ para n^2.

Telgarsky (2016) [52] vai além das analogias e mostra que para qualquer inteiro positivo k, existem redes neurais com $\Theta(k^3)$ camadas, cada qual com $\Theta(1)$ nós e $\Theta(1)$ parâmetros distintos, que não podem ser aproximadas por redes neurais com $O(k)$ camadas, a não ser que essas tenham uma quantidade de neurônios exponencialmente grande, da ordem de $\Omega(2^k)$ neurônios. Esse resultado é provado para portas semi-algébricas, as quais incluem funções como ReLU, operadores de máximo (como *maxpooling*), e funções po-

linomiais por partes. Isso permitiu estabelecer o papel fundamental da profundidade tanto em redes neurais convencionais quanto convolucionais que utilizam ReLU e operações de máximo. Abaixo a reprodução do teorema.

Teorema 3.1. *Seja qualquer inteiro $k \geq 1$ e qualquer dimensão $d \geq 1$. Existe uma função $f : \mathbb{R}^d \to \mathbb{R}$ computada por uma rede neural com portas ReLU em $2k^3 + 8$ camadas, com $3k^3 + 12$ neurônios no total, e $4 + d$ parâmetros distintos de forma que:*

$$\inf_{g \in \mathscr{C}} \int_{[0,1]^d} |f(x) - g(x)| dx \geq \frac{1}{64},$$

onde $\mathscr{C}$ é a união dos seguintes dois conjuntos de funções:
(1) funções computadas por redes compostas por portas (t, α, β)-semi-algébricas com número de camadas $\leq k$ e número de neurônios $2^k/(t\alpha\beta)$ — como é o caso de redes que utilizam funções ReLU ou redes convolucionais com funções ReLU e portas de maximização;
(2) funções computadas por combinações lineares de um número de árvores de decisão $\leq t$ com $2^{k^3}/t$ neurônios — como as funções utilizadas em boosted decision trees.

O nome porta semi-algébrica vem de conjuntos semi-algébricos, definidos por uniões e intersecções de desigualdades polinomiais, e denota uma função que mapeia algum domínio em $\mathbb{R}$. Em redes neurais o domínio da função deve ser particionado em três argumentos: o vetor de entrada entrada, o vetor de parâmetros, e o vetor de números reais advindos de neurônios predecessores.

A prova do Teorema é feita em três etapas, demonstrando: (1) que funções com poucas oscilações aproximam de forma pobre funções com mais oscilações, (2) que funções computadas por redes neurais com menos camadas possuem menos oscilações, (3) que funções computadas por redes com mais camadas podem ter mais oscilações. A prova vem do teorema de hierarquia de profundidade para circuitos booleanos do trabalho seminal de Håstad [22] que estabelece não ser possível aproximar funções de paridade de circuitos profundos por circuitos menos profundos a não ser que o tamanho desses últimos seja exponencial. Por questões de espaço recomendamos a leitura do artigo completo de Telgarsky [52].

Propriedades de contração e separação Segundo Mallat (2016) [34], o funcionamento das CNN está ligado à eliminação de elementos/variáveis não informativas, via contração ou redução da dimensão espacial nas direções mais apropriadas a um certo problema. O autor define o problema de aprendizado supervisionado, que computa uma aproximação $\hat{f}(\mathbf{x})$ de uma função $f(\mathbf{x})$ por meio de um conjunto de treinamento $\mathbf{X} \in \Omega$, sendo que o domínio de Ω é um subconjunto aberto de alta dimensionalidade de $\mathbb{R}^d$ (e não um manifold de baixa dimensão). Então os conceitos de separação e linearização são apresentados.

Idealmente, deseja-se reduzir a dimensão de um vetor de entrada $\mathbf{x}$ computando um vetor de menor dimensão $\phi(\mathbf{x})$. Assim, $\phi()$ é um operador de contração que reduz o intervalo de variações de $\mathbf{x}$ e que, ao mesmo tempo, separa diferentes valores da função f (classificador ou regressor), ou seja, $\phi(\mathbf{x}) \leq \phi(\mathbf{x}')$ se $f(\mathbf{x}) \leq f(\mathbf{x}')$. Assim, ϕ **separa**

f, encontrando uma projeção linear de $\mathbf{x}$ em algum espaço $\mathcal{V}$ de menor dimensão k, que separa f.

Como estratégia alternativa, as variações de f podem ser linearizadas de forma a reduzir a dimensionalidade. Nesse cenário um projetor linear de baixa dimensionalidade de $\mathbf{x}$ pode separar os valores de f se essa permanecer constante na direção do espaço linear de alta dimensão. Após encontrar $\phi(\mathbf{x})$ que realize essa linearização mantendo $f(\mathbf{x})$ constante, a dimensão é reduzida aplicando projeção linear em $\phi(\mathbf{x})$.

Redes neurais profundas progressivamente contraem o espaço e lineariza transformações ao longo das quais f permanece aproximadamente constante, de forma a preservar a separação. A combinação de canais (feature maps) provêm a flexibilidade necessária para estender translações para maiores grupos de simetrias locais. Assim, as redes são estruturadas de forma a fatorizar grupos de simetrias, em que todos os operadores lineares são convoluções.

Assim como em [32], Mallat [34] faz conexões com conceitos de física, demonstrando que redes hierárquicas multiescala computam convoluções de forma que as classes sejam preservadas. Os pesos geram repostas fortes a padrões particulares e são invariantes a uma série de transformações e os filtros são adaptados para produzir representações esparsas de vetores de suporte multiescala. Esses vetores proveriam então um código distribuído que define uma memorização de padrões invariante. A interpretação de geometria diferencial é a de que os pesos são transportados ao longo de um maço de fibras (conceito matemático que estaria relacionado a atualização dos filtros). Conforme a contração aumenta, aumenta o número de vetores de suporte necessários para realizar a separação o que indicaria um incremento também no número de fibras permitindo explicar simetrias e invariâncias suficientemente estáveis para a obtenção de transferência de aprendizado. Isso possibilita importante interpretação do aprendizado e abre espaço para pesquisas teóricas na área.

3.5.1. Limitações de Deep Learning e Considerações Finais

Técnicas de Deep Learning tem conseguido atingir resultados estatisticamente impressionantes em particular, como demonstrado, pelo uso de múltiplas camadas. Entretanto, existem limitações no uso de redes neurais profundas pois essas são basicamente uma forma de aprender uma série de transformações a serem aplicadas ao vetor de entrada. Essas transformações são dadas por um grande conjunto de pesos (parâmetros) que são atualizados durante a etapa de treinamento de forma a minimizar a função de custo. A primeira limitação para que seja possível realizar o treinamento, é que tais transformações precisam ser deriváveis, isso é, o mapa entre a entrada e saída da rede deve ser contínuo e idealmente suave [6].

A segunda limitação diz respeito à abstração e adaptação: redes neurais profundas precisam de quantidades massivas de dados rotulados para aprender conceitos simples; em contrapartida, seres humanos são capazes de aprender um conceito a partir de pequenas quantidades de exemplos. No problema de classificação de dígitos, é necessário fornecer milhares de exemplos de dígitos, escritos por pessoas diferentes para que a rede possa aprender a diferenciá-los, enquanto um humano seria capaz de aprender os mesmos conceitos com alguns poucos exemplos escritos por uma única pessoa. Humanos também

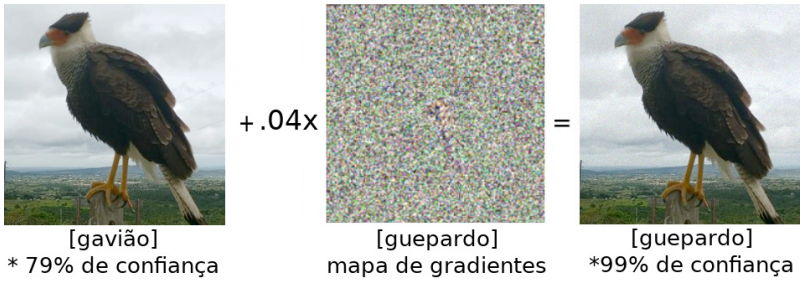


Figura 3.7. Imagem gerada pela adição de 4% do gradiente de features de guepardo em uma imagem de um gavião, que é então classificada como guepardo com alta confiança pela CNN.

conseguem adaptar facilmente esses conceitos com o passar do tempo.

Mesmo considerando tarefas a princípio solucionáveis via Deep Learning, existem diversos estudos mostrando falhas que fazem duvidar da real capacidade de generalização desses modelos, em particular: (i) perturbações nos dados de entrada, tais como ruído, podem impactar significativamente os resultados [37], (ii) o uso de imagens invertidas na fase de teste faz com que as redes errem completamente os resultados [21], (iii) é possível criar imagens que visualmente parecem conter somente ruído aleatório, mas que fazem com que CNNs as classifiquem com 99% de certeza [38], (iv) é possível desenvolver perturbações imperceptíveis a humanos (em torno de 4%) que fazem redes produzirem saídas completamente erradas [39], conforme mostramos na Figura 3.7, um exemplo adaptado adicionando informações do gradiente de de uma imagem de guepardo a uma imagem de um falcão, o que faz com que a imagem seja classificada com alta confiança como guepardo por uma CNN do estado da arte. Para aliviar o impacto dessa fragilidade dos métodos de Deep Learning a ataques, estudos recentes recomendam a inclusão de exemplos adversários no conjunto de treinamento ou o aumento da capacidade (em termos de filtros/neurônios por camada), aumentando o número de parâmetros e assim a complexidade do espaço de funções permitidas pelos modelo [45],[33].

Ao analizar os cenários citados do ponto de vista da Teoria do Aprendizado Estatístico [53], podemos criar a hipótese de que esses métodos estão de fato aprendendo um modelo baseado em memória. Eles funcionariam bem em diversos casos pois são treinados em milhões de exemplos, fazendo com que imagens nunca vistas se encaixem nas informações memorizadas. Estudos recentes incluem a análise tensorial de representações [7] e incertezas [12] para tentar esclarecer o comportamento do aprendizado das redes neurais profundas. Ainda [34] e [31] publicaram resultados teóricos importantes, com interpretações que podem dar novas direções à compreensão e ao estudo das garantias teóricas e limites, mas ainda há muito a percorrer do ponto de vista do embasamento teórico relacionado ao aprendizado supervisionado e não supervisionado utilizando redes profundas.

Para maiores detalhes, aplicações, e outros modelos de redes incluindo Autoencoders e Modelos Gerativos sugerimos a leitura do survey recente sobre Deep Learning em Visão Computacional [44], bem como o livro de Goodfelow et al. disponível online [14].

É inegável o impacto positivo de Deep Learning para diversas áreas em que o aprendizado de máquina é empregado para busca de soluções. Compreendendo suas limi-

tações e funcionamento, é possível continuar a explorar esses métodos e alcançar avanços notáveis em diversas aplicações.

Agradecimentos

Os autores agradecem o apoio da FAPESP (processos #2016/16411-4 e #2015/05310-3)

Referências

[1] Shai Ben-David and Shai Shalev-Shwartz. *Understanding Machine Learning: from theory to algorithms*. Cambridge, 2014.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, August 2013.

[3] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838*, 2016.

[4] T Bui, L Ribeiro, M Ponti, and John Collomosse. Compact descriptors for sketch-based image retrieval using a triplet loss convolutional neural network. *Computer Vision and Image Understanding*, 2017.

[5] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference (BMVC)*, page 1405.3531, 2014.

[6] F. Chollet. *Deep Learning with Python*. Manning, 2017.

[7] Nadav Cohen, Or Sharir, and Amnon Shashua. On the expressive power of deep learning: A tensor analysis. In *Conference on Learning Theory*, pages 698–728, 2016.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[9] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[10] Asja Fischer and Christian Igel. Training restricted boltzmann machines: An introduction. *Pattern Recognition*, 47(1):25–39, 2014.

[11] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988.

[12] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

[13] Rafael Gonzalez and Richard Woods. *Digital Image Processing*. Pearson, 3 edition, 2007.

[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[15] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, 2014.

[16] P. Goyal, P. Dollar, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch SGD: Training imagenet in 1 hour.

[17] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649, 2013.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[20] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[21] Hossein Hosseini and Radha Poovendran. Deep neural networks do not recognize negative images. *arXiv preprint arXiv:1703.06857*, 2017.

[22] Johan Håstad. *Computational Limitations of Small Depth Circuits*. PhD thesis, MIT, 1986.

[23] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[24] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*, 2015.

[25] AN Kolmogorov. The representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables. *Doklady Akademii Nauk SSSR*, 108(2):179–182, 1956.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems.*, pages 1106–1114, 2012.

[27] Hugo Larochelle, Michael Mandel, Razvan Pascanu, and Yoshua Bengio. Learning algorithms for the classification restricted boltzmann machine. *Journal of Machine Learning Research*, 13(Mar):643–669, 2012.

[28] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[29] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[30] Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670. ACM, 2014.

[31] Henry W Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, pages 1–25, 2016.

[32] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):2024–2039, 2016.

[33] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[34] Stéphane Mallat. Understanding deep convolutional networks. *Phil. Trans. R. Soc. A*, 374(2065):20150203, 2016.

[35] H.N. Mhaskar and T. Poggio. Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*, 14(06):829–848, 2016.

[36] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814. Omnipress, 2010.

[37] T. Nazare, G. Paranhos da Costa, W. Contato, and M. Ponti. Deep convolutional neural networks and noisy images. In *Iberoamerican Conference on Pattern Recognition (CIARP)*, 2017.

[38] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436, 2015.

[39] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[40] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[41] Moacir Ponti, Elias S Helou, Paulo Jorge SG Ferreira, and Nelson DA Mascarenhas. Image restoration using gradient iteration and constraints for band extrapolation. *IEEE Journal of Selected Topics in Signal Processing*, 10(1):71–80, 2016.

[42] Moacir Ponti, Josef Kittler, Mateus Riva, Teófilo de Campos, and Cemre Zor. A decision cognizant Kullback–Leibler divergence. *Pattern Recognition*, 61:470–478, 2017.

[43] Moacir Ponti, Tiago S Nazaré, and Gabriela S Thumé. Image quantization as a dimensionality reduction procedure in color and texture feature extraction. *Neurocomputing*, 173:385–396, 2016.

[44] Moacir Ponti, Leonardo S.F. Ribeiro, Tiago S. Nazare, Tu Bui, and John Collomosse. Everything you wanted to know about deep learning for computer vision but were afraid to ask. In *SIBGRAPI — Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T 2017)*, pages 1–26, 2017.

[45] Andras Rozsa, Manuel Günther, and Terrance E Boult. Are accuracy and robustness correlated. In *Machine Learning and Applications (ICMLA), 2016 15th IEEE International Conference on*, pages 227–232. IEEE, 2016.

[46] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[47] Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Artificial Intelligence and Statistics*, pages 448–455, 2009.

[48] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[49] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR (workshop track)*, 2015.

[50] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284, 2017.

[51] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[52] Matus Telgarsky. Benefits of depth in neural networks. *arXiv preprint arXiv:1602.04485*, 2016.

[53] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[54] David Warde-Farley, Ian J Goodfellow, Aaron Courville, and Yoshua Bengio. An empirical analysis of dropout in piecewise linear networks. In *ICLR 2014*, 2014.

[55] Hugh E Warren. Lower bounds for approximation by nonlinear manifolds. *Transactions of the American Mathematical Society*, 133(1):167–178, 1968.

[56] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[57] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip Torr. Conditional random fields as recurrent neural networks. In *International Conference on Computer Vision (ICCV)*, 2015.

Sobre os Autores

Moacir A. Ponti é professor Doutor no Instituto de Ciências Matemáticas e de Computação (ICMC) da Universidade de São Paulo (USP), em São Carlos/SP, Brasil. Possui Doutorado (2008) e Mestrado (2004) pela Universidade Federal de São Carlos. Foi pesquisador visitante no Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, onde realizou estágio pós-doutoral em 2016. Coordenador de projetos de pesquisa nas áreas de Visão Computacional e Reconhecimento de Padrões financiado por agências públicas (FAPESP, CNPq), institutos e empresas (Google, UGPN). Autor de mais de 40 artigos publicados em conferências e periódicos com revisão por pares. Atua nas áreas de Processamento de Sinais, Imagens e Videos, com ênfase em problemas inversos, aprendizado e extração de características com aplicações.

Gabriel B. Paranhos da Costa é doutorando no Instituto de Ciências Matemáticas e de Computação (ICMC) da Universidade de São Paulo (USP), em São Carlos/SP, Brasil, com período sanduíche na Universidade de Edimburgo, Reino Unido. Possui Bacharelado em Ciências da Computação (2012) e Mestrado (2014) pelo ICMC/USP. Realiza pesquisas na área de Processamento de Imagens e Reconhecimento de Padrões, tendo como principal foco detecção de anomalias e aprendizado e extração de características para aplicações de aprendizado de máquina.

REALIZAÇÃO

SBC Sociedade Brasileira de Computação

ORGANIZAÇÃO

UFU

Faculdade de Computação

APOIO

ProPP

UBERLÂNDIA CONVENTION & VISITORS BUREAU

FOMENTO

CAPES

FAPEMIG

CNPq

PATROCÍNIO DIAMANTE

Itaú

NVIDIA

PATROCÍNIO OURO

Algar Telecom

PATROCÍNIO PRATA

Google

IBM Research | Brasil

PATROCÍNIO BRONZE

SEBRAE

neppo

SANKHYA GESTÃO DE NEGÓCIOS

ClickPerformance®