Capítulo 4

Análise de Dados em Redes Sem Fio de Grande Porte: Processamento em Fluxo em Tempo Real, Tendências e Desafios

Dianne S. V. Medeiros (UFF), Helio N. C. Neto (UFF),
Martin Andreoni Lopez (Samsung R&D), Luiz Claudio S. Magalhães (UFF),
Edelberto F. Silva (UFJF), Alex B. Vieira (UFJF),
Natalia C. Fernandes (UFF), Diogo M. F. Mattos (UFF)

Abstract

In this chapter, we focus on knowledge extraction from large wireless networks through stream processing. We present the primary methods of sampling, data collection and monitoring of wireless networks and we characterize knowledge extraction as a machine learning problem on big data stream processing. The Apache Spark and Apache Flink are the main trends on big data stream processing frameworks and, thus, are discussed in this chapter. We explore the data preprocessing, the feature engineering and the machine learning algorithms applied to the scenario of wireless network analytics. We address challenges and research projects in wireless network monitoring and stream processing. Finally, future perspectives, such as deep learning and reinforcement learning in stream processing, are anticipated.

Resumo

Este capítulo foca na extração de conhecimento em aplicações de redes sem fio de grande porte através do processamento de dados em fluxo. O capítulo apresenta os principais métodos de amostragem, coleta de dados e monitoramento de redes sem fio e caracteriza a extração do conhecimento como um problema de aprendizado de máquina em processamento de grandes massas de dados em fluxo. As plataformas Apache Spark e Apache Flink são as principais tendências entre plataformas para o processamento de dados em fluxo e, portanto, são analisadas neste capítulo. Discute-se o pré-processamento, a engenharia de características e os algoritmos para o aprendizado de máquina aplicados ao cenário de redes sem fio. Desafios e projetos de pesquisa em monitoramento de redes sem fio e processamento em fluxo são elencados. Por fim, perspectivas futuras, como a aprendizagem profunda e por reforço aplicadas a dados em fluxo, são antecipadas.

4.1. Introdução

A popularização de celulares inteligentes e dispositivos conectados à Internet das Coisas (*Internet of Things* - IoT) [Mattos et al., 2018] impulsionou o crescimento da geração de dados provenientes de dispositivos móveis em redes sem fio. Nos últimos 5 anos, as redes móveis alcançaram um crescimento acumulado de dezoito vezes, registrando 63% de aumento no volume de dados trafegado no ano de 2016 em relação ao ano anterior [Cisco, 2017]. O crescente volume de tráfego nas redes móveis também impacta redes sem fio fixas, já que a terceirização do tráfego de dados de dispositivos móveis para redes sem fio fixas compõe 60% do tráfego sem fio total. Assim, as redes IEEE 802.11 representam o principal meio de acesso de uma parcela significativa dos usuários finais, nos mais variados ambientes. Em universidades e em empresas, grande parte dos usuários conecta à Internet através da rede sem fio institucional ou a serviços internos através de intranet sem fio. A rede sem fio institucional da Universidade Federal Fluminense (UFF), por exemplo, conta com a operação de 547 pontos de acesso que atendem a uma população de mais de 60 mil usuários, com picos de 5 mil usuários conectados concomitantemente, gerando mais de 100 Mb/s de dados a serem analisados [Magalhães e Mattos, 2018]. Os dados de gerenciamento e monitoramento dessas redes contêm conhecimento sobre os usuários, as redes e os padrões de uso e de deslocamento [Divgi e Chlebus, 2013]. Assim, as redes sem fio tornam-se excelentes fontes geradoras de dados.

Alinhado a essa tendência, há o crescente interesse em medições do desempenho fim a fim e seu impacto em aplicações móveis [Goel et al., 2015]. O monitoramento das redes móveis oferece informações úteis para pesquisadores e traz benefícios aos operadores de redes. A ausência de mecanismos de inteligência e de ação rápida na rede, muitas vezes associada a visões limitadas de ferramentas de controle por fluxo, prejudicam a qualidade de experiência (QoE) de usuários de redes sem fio de grande escala e, em especial, a extração de conhecimento sobre os usuários e as redes [Jang et al., 2017].

A popularidade das redes sem fio fomenta, então, o desafio de processar e gerenciar o grande volume de dados gerado, já que um único ponto de acesso suporta, tipicamente, algumas dezenas de clientes conectados [Magalhães e Mattos, 2018]. O monitoramento de redes sem fio também apresenta diversos desafios quando comparado ao monitoramento de redes cabeadas, tornando a tarefa ainda mais complexa. A replicação dos métodos usados tradicionalmente, realizando medidas dos parâmetros após os dados passarem para a rede cabeada, não revela o estado real da rede sem fio. Os métodos atuais não permitem distinguir, por exemplo, uma rede ociosa de uma que está tão congestionada que nenhum quadro está sendo entregue. Propostas para obter o estado das redes consideram medidas ativas, mas implicam alterações nos parâmetros avaliados. Medidas indiretas, como medir o uso do canal através dos contadores dos próprios pontos de acesso ou usar sensores para análise espectral e pontos para captura e análise de quadros, são empregadas ao custo de informações menos precisas. Além disso, metadados coletados nas redes permitem a criação de aplicações que tornam o ambiente ciente de seu contexto, auxiliando no monitoramento em todos os níveis. É válido ressaltar que a configuração de uma rede sem fio de larga escala é uma atividade complexa, tanto pelo número de elementos a serem configurados, como pelos fatores de propagação rádio e interferências, que não são simples de serem estimados *a priori*. Assim, o monitoramento é uma ferramenta chave para entender a topologia *de facto* da rede. A captura dos *beacons*

recebidos através de varredura passiva, associada à potência com que foram recebidos, permite identificar quais pontos de acesso são vizinhos e a distância rádio entre eles. A coleta de dados permite ainda a execução de operações básicas em redes, como cobranças de serviços, detecção de ameaças, isolamento e mitigação de falhas.

As redes móveis sem fio fornecem também informação espaço-temporal sobre usuários e condições da rede para dotar o sistema de visibilidade e inteligência fim a fim, permitindo melhor compreensão da dinâmica da rede a longo prazo. Informações sobre geolocalização [Cici et al., 2015] ou sobre o posicionamento dos usuários [Alessandrini et al., 2017] permitem identificar padrões de uso e detectar anomalias. Além disso, a análise dos dados fornecidos por essas redes habilita a auto-coordenação das funções e entidades da rede e a construção de redes mais eficientes e proativas. A análise do grande volume de dados provenientes das redes sem fio é desafiadora devido às características inerentes à própria rede, como a mobilidade dos usuários, o ruído presente e a redundância dos dados coletados que impactam diretamente as cinco dimensões fundamentais do processamento de grande massas de dados, volume, velocidade, variedade, valor e veracidade. Nesse contexto, algumas técnicas tradicionais de processamento de grandes massas de dados podem ser usadas, mas é necessário empregar técnicas de processamento de dados em fluxo para análise da rede em tempo real.

O processamento de grandes massas de dados em fluxo consiste no tratamento de dados potencialmente não limitados em número de amostras, isto é, que chegam a todo momento, e não limitados quanto a espaço de atributos, ou seja, o universo de atributos dos dados assim como a distribuição estatística são desconhecidos. A ideia de processamento em fluxo contrapõe-se ao processamento em lotes, em que um conjunto de dados bem limitado e conhecido é processado de uma só vez por uma plataforma de processamento de grande massa de dados. O processamento em lote exige grande disponibilidade de memória para armazenamento dos dados e implica maior latência na geração de respostas de processamento. A principal característica do processamento de dados em fluxo frente ao processamento em lotes é a menor latência no tempo de processamento de cada amostra de dados. Contudo, o processamento em fluxo impõe restrições de memória para o armazenamento dos dados entrantes e requer que algoritmos tradicionais de aprendizado de máquina sejam adaptados ao cenário em que o conjunto de dados de treinamento não pode ser revistado devido ao número ilimitado de amostras e ao requisito de latência mínima ao processar cada nova amostra. Para tanto, plataformas de processamento de dados em fluxo, como Apache Spark Streaming [Zaharia et al., 2013] e Apache Flink [Carbone et al., 2015b], propõem dois modelos distintos de processamento de amostras em fluxo, o processamento em micro-lotes e o processamento por amostra.

Ao adotar mecanismos de aprendizado de máquina para extrair conhecimento de dados em fluxo, os algoritmos estão sujeitos a erros na aprendizagem em reação a mudanças de conceito nos dados de entrada [Andreoni Lopez et al., 2019]. Nesse sentido, aplicações de aprendizado de máquina devem ser conscientes da distribuição estatística dos atributos dos dados de entrada e devem verificar quando há uma mudança nas distribuições. As aplicações de aprendizado de máquina para processamento de dados em fluxo podem considerar que os dados entrantes são rotulados por algum processo que gere uma verdade básica, ou que existe um modelo de aprendizado previamente definido ou, ainda, podem buscar padrões ocultos nos dados sem que nenhum modelo seja fornecido.

O objetivo deste capítulo é apresentar os principais algoritmos e técnicas de processamento em fluxo, *stream processing*, em tempo real para extração de conhecimento do monitoramento de redes sem fio em grande escala, chamado de *Wi-Fi Analytics*. É apresentada uma visão teórica do processamento de dados em fluxo e do uso de algoritmos de aprendizado de máquina com treinamento em tempo real. Além disso, são discutidas algumas aplicações dos algoritmos no monitoramento de redes sem fio. O monitoramento das redes sem fio consiste tanto na caracterização do tráfego gerado pelos usuários do serviço quanto na avaliação espectral em tempo real. Tais vertentes de monitoramento geram grandes massas de dados, que devem ser processadas em tempo real, e o resultado do processamento permite identificar padrões de uso, definir perfis de usuários, identificar falhas ou queda de desempenho em pontos específicos da rede ou otimizar alocações de canais. Nesse sentido, o monitoramento e o controle de redes sem fio de grande porte compõem um cenário que exige o processamento em tempo real e de respostas imediatas para a adequação da rede a picos de uso e concentrações esporádicas de usuários. Assim, no cenário de análise de dados em redes sem fio de grande porte é essencial o uso dos algoritmos de processamento em fluxo.

Este capítulo está organizado da seguinte forma. A Seção 4.2 apresenta ferramentas e métodos para a gerência e o monitoramento de redes sem fio. A Seção 4.3 define as etapas para a realização da extração de conhecimento em grandes massas de dados. As plataformas de processamento de dados em fluxo Apache Spark e Apache Flink são exploradas na Seção 4.4. O aprendizado de máquina em fluxo e algoritmos de aprendizado incremental são detalhados na Seção 4.5. A Seção 4.6 elenca os desafios de pesquisa e as perspectivas futuras na análise de grandes massas de dados gerados por redes sem fio. Por fim, a Seção 4.7 conclui o capítulo.

4.2. Monitoramento de Redes Sem Fio

Grandes quantidades de dados geradas a partir de dispositivos móveis e que não podem ser processadas em uma única máquina constituem o conceito de *Mobile Big Data* (MBD) [Guo et al., 2018]. A Figura 4.1 mostra um esquema das camadas que compõem o MBD. Na primeira, Fontes, ocorre a geração de dados a partir da coleta por parte de dispositivos móveis utilizando tecnologias como Wi-Fi, GPS e comunicação celular. Na segunda camada, Integração, os dados das diferentes fontes heterogêneas são integrados. Primeiramente, os dados são validados e em seguida são unificados. Para esse fim, os dados são pré-processados, usando diferentes técnicas aplicadas para a limpeza e integração. Na terceira camada, Preparação, o pré-processamento dos dados continua, preparando-os para o consumo na camada de analíticas. Na quarta camada, Analíticas, a análise dos dados é feita através de diferentes algoritmos para identificação de padrões, classificação, entre outros. Finalmente, na quinta camada, Aplicação, os resultados produzidos pela quarta camada são usados por diferentes aplicações.

Trabalhos no contexto de MBD focam na localização de objetos combinando o uso de aprendizado de máquina para redes sem fio. Essas técnicas compõem a Ciência de Análise do Wi-Fi (*Wi-Fi Analytics*), e utilizam pacotes de gerenciamento, como pedido/resposta de *probes* e *beacons* do protocolo IEEE 802.11 para determinar a localização e deslocamento de objetos [Acer et al., 2015, Acer et al., 2016]. Os algoritmos utilizados para estimar localização são as Máquinas de Vetores de Suporte (*Support Vector*

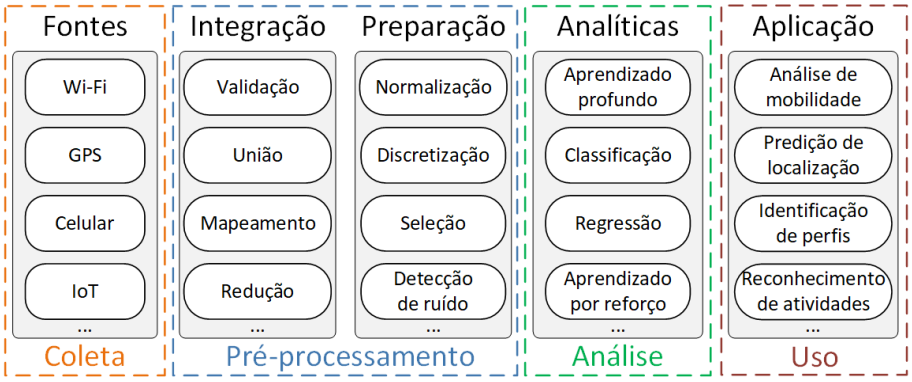


Figura 4.1. Esquema do processo de *Mobile Big Data*. A primeira camada gera os dados, enquanto a segunda integra os dados provenientes de diferentes fontes. Na terceira e quarta camadas esses dados são pré-processados para serem consumidos pela quinta camada, na qual diferentes algoritmos são aplicados para analisar os dados. Por fim, diferentes aplicações fazem uso dos resultados das análises realizadas.

Machine - SVM), com *kernel* gaussiano. As informações físicas, como intensidade do sinal contidas nos *probes*, são obtidas dos *gateways* Wi-Fi, podendo estimar, por exemplo, a movimentação de multidões. A mobilidade deduzida a partir de grandes massas de dados em geral combina diferentes sinais sem fio, como serviços de posicionamento global (GPS), sinais celulares como GSM e LTE, sinais Wi-Fi, entre outros, para determinar a geolocalização de pessoas e as trajetórias seguidas de forma acurada [Xu et al., 2017]. Nesse contexto, é possível também determinar a localização e as trajetórias de veículos, de forma que sistemas de tráfego inteligente que utilizam ferramentas de *Big Data* podem analisar dados provenientes de diferentes fontes sem fio para mitigar problemas de engarrafamento [Chen et al., 2018]. Não havendo dados provenientes de outras redes sem fio além da rede Wi-Fi, a acuidade da localização de objetos e pessoas dependerá da densidade de pontos de acesso existentes na rede. Quanto menos densa a rede, menor é a acuidade da localização encontrada. Os dados de mobilidade coletados são analisados por aprendizagem profunda, classificação e regressão para adquirir conhecimento útil como a previsão do fluxo de tráfego de veículos. Os algoritmos utilizados são os modelos de séries temporais, previsão baseada em aprendizado profundo, modelos de cadeias de Markov e combinação de redes neurais (*Neural Networks* - NNs) e média móvel integrada autorregressiva (*Autoregressive Integrated Moving Average* - ARIMA).

A ciência de análise do Wi-Fi também pode ser usada para melhorar a qualidade de experiência (*Quality of Experience* - QoE) dos usuários. Moura *et al.* apresentam uma combinação de aprendizado de máquina com as redes definidas por software (*Software Defined Networking* - SDN) para monitorar e controlar o congestionamento dos canais das redes locais sem fio (*Wireless Local Area Networks* - WLANs) [Moura et al., 2019]. A técnica de aprendizado de máquina utilizada é o bandido de k-braços (*Multi-Armed Bandit*), que é um algoritmo de aprendizado por reforço. O algoritmo funciona como realimentação para o controlador SDN para melhorar a experiencia do usuário no acesso de páginas *Web*. A ciência de análise do Wi-Fi no contexto de MDB permite, então, caracterizar as redes sem fio IEEE 802.11, através da identificação dos atores e fatores presentes no ambiente a partir de dados coletados. Essa caracterização pode estabelecer desde o posicionamento de usuários em ambientes *indoor* e *outdoor*, até a identificação de

pontos de acesso e redes concorrentes do meio a partir da gerência do espectro. Diversas abordagens podem ser seguidas para esse fim, como a caracterização da topologia da rede através do monitoramento de pontos de acesso e a caracterização do espectro. Em todo caso, é necessário utilizar ferramentas específicas para coletar os dados.

4.2.1. Caracterização de redes sem fio

O ambiente no qual as redes sem fio estão imersas está em constante mudança. Apesar dos pontos de acesso que formam a infraestrutura da rede normalmente estarem fixos e conectados à estrutura da rede cabeada, a mobilidade dos usuários causa flutuação no nível de ruído ao qual os pontos de acesso estão submetidos e interferem no ambiente rádio, já que os seres humanos são uma barreira para a propagação de micro-ondas. Em redes de grande porte, novos pontos de acesso podem ser ligados e desligados a qualquer tempo, alguns pontos de acesso podem ser dotados de mobilidade devido à proliferação do compartilhamento da Internet celular via Wi-Fi e o ambiente rádio pode ser modificado ainda pela abertura e fechamento de portas ou outras alterações do meio físico. Dessa forma, a caracterização da rede se torna uma tarefa complexa. Uma primeira abordagem para caracterizar a rede é realizar um estudo de vizinhança. Uma das formas de identificar a vizinhança de um ponto de acesso é coletar os *beacons* recebidos [Magalhães e Mattos, 2018]. Todo ponto de acesso envia quadros *beacon* periodicamente. O *beacon* faz parte da sincronia da rede sem fio, e de vários mecanismos, como os de economia de energia. Por isso nenhum ponto de acesso pode suprimir o seu envio, apesar de ser possível não incluir o nome da rede nesses quadros. Capturar os *beacons* permite, então, a descoberta de todos os pontos de acesso em uma área. Essa é uma forma de varredura passiva, mas pode ser estimulada pelo envio de quadros de `probe-request`, que gera a criação e envio de quadros `probe-response`. Seja a varredura passiva ou estimulada, esse tipo de coleta de dados identifica apenas os pontos de acesso.

A medida da potência recebida em um *beacon* é um indicador da distância rádio do emissor ao receptor. Como o ambiente varia continuamente, uma única medida não é confiável, mas com várias medidas é possível criar um mapa confiável das distâncias rádio entre pontos de acesso. Em uma rede sem fio institucional essas medidas podem ser usadas para criar uma configuração dos canais e da potência para cada ponto de acesso a fim de minimizar a interferência entre pontos de acesso vizinhos, sejam eles pertencentes à rede ou gerenciados por terceiros [Magalhães e Mattos, 2018]. Os pontos de acesso pertencentes à rede institucional têm conhecimento das estações associadas a eles e podem trocar essas informações com um sistema de monitoramento centralizado. Dessa forma, é possível para um operador saber o número de usuários associados a cada ponto de acesso e o número global de usuários associados à rede. No entanto, uma rede sem fio geralmente não tem conhecimento das estações associadas a outra rede sem fio.Para saber quantos usuários estão associados a redes vizinhas é necessário capturar pacotes para identificar os MACs de origem. Para inferir interferências que não sejam causadas por redes IEEE 802.11, tal como as causadas por dispositivos bluetooth, telefones sem fio ou fornos de micro-ondas, são necessários mecanismos como a varredura espectral.

O conhecimento da topologia da rede além de permitir explorar características relacionadas ao ambiente rádio, permite obter informações sobre os usuários. A localização de um usuário pode ser feita em baixa resolução assumindo que ele se encontra na vizi-

nhança do ponto de acesso ao qual está associado. De forma geral, como uma estação fica associada a apenas um ponto de acesso a cada instante, é possível usar o histórico ou sequência de associações a pontos de acesso para melhorar a acurácia da localização. No entanto, devido aos algoritmos usados atualmente para a escolha de pontos de acesso causarem muitas trocas mesmo quando o usuário está estático [Balbi et al., 2012], devido às flutuações da potência de sinal recebida provocadas pela variabilidade do ambiente de rádio, o histórico de associações apresenta precisão limitada. Assim, é necessário monitorar os pontos de acesso para descobrir a topologia da rede, caracterizar o ambiente rádio e os usuários associados à rede.

Monitoramento de pontos de acesso

Redes sem fio de grande porte necessitam de um controle centralizado para tomada de decisões, como a utilização do espectro e a alocação de clientes. O meio sem fio pode sofrer interferência de diversas fontes e, nesse ponto, a alocação de canais é um desafio. A complexidade do problema cresce ao se adicionar a restrição de prover um desempenho aceitável para usuários na sobreposição de canais [Luiz et al., 2015]. Canais próximos geram interferências e diminuem o desempenho das redes sem fio em suas vizinhanças. Portanto, é necessário que o projeto de instalação de redes sem fio considere a alocação de canais tanto do ponto de acesso instalado, como dos demais pontos de acesso já presentes na área. A alocação de canais ótima é um problema NP-Difícil [Leung e Kim, 2003], mas heurísticas alcançam boas soluções de configuração de canais [Maturi et al., 2017].

Em redes sem fio de grande porte, alternativas proprietárias para o gerenciamento de pontos de acesso, e consequente atribuição de canais, têm bom desempenho com a contrapartida de um alto custo financeiro [Balbi et al., 2012]. Para evitar esse elevado custo, *Balbi et al.* propõem um algoritmo para alocação de canais que considera primeiro a interferência interna, entre os pontos de acesso controlados, e depois a interferência externa, dos pontos de acesso com todos os pontos de acesso nas vizinhanças. Maturi *et al.* propõem um esquema de seleção dinâmica de canais que permite que uma rede IEEE 802.11 salte sobre os canais disponíveis sempre escolhendo o que tem menor utilização [Maturi et al., 2017]. Os autores implementam a proposta e mostram a viabilidade de realizar o salto em frequências para redes IEEE 802.11 com equipamento padrão de mercado. Outras propostas realizam a alocação de canal através da interferência observada entre pontos de acesso [Lin et al., 2017, Monteiro et al., 2016] e entre clientes e pontos de acesso [Luiz et al., 2015]. É importante também considerar o impacto do *handoff* nas redes sem fio de grande porte na qualidade da rede percebida pelo usuário. Nesse contexto, Shin *et al.* descrevem a rede sem fio através de um grafo e investigam como diminuir a latência durante o *handoff* [Shin et al., 2004]. A ideia central é desenvolver algoritmos que permitam que o *handoff* ocorra sem que a estação tenha a necessidade de ficar monitorando todos os canais. Para tanto, a estação armazena o conjunto de canais que cada vizinho está operando e o conjunto de pontos de acesso vizinhos em cada canal.

Huang *et al.* utilizam plataformas de MBD, ferramentas de análise de dados e aquisição distribuída para monitoramento de pontos de acesso sem fio na rede 3G WCDMA da Unicom, na China [Huang et al., 2014]. Os autores utilizam uma plataforma de armazenamento e análise de dados baseada no Sistema de Arquivos Distribuídos Hadoop (*Hadoop Distributed File System* - HDFS). Uma arquitetura semelhante à utilizada

pelos autores pode ser empregada para processar os dados de outras redes sem fio de grande porte, como redes Wi-Fi institucionais. Hadi *at al.* apresentam diversos trabalhos relacionados ao tema [Hadi et al., 2018].

Caracterização do ambiente rádio

As redes sem fio de próxima geração, como o 5G, devem suportar taxas de transmissão de dados extremamente altas e paradigmas de aplicações diversos e novos, que exigirão novas tecnologias rádio sem fio. O desafio é ajudar a rede sem fio no aprendizado adaptativo e inteligente na tomada de decisões, de modo que os diversos requisitos dessas redes possam ser atendidos. O aprendizado de máquina é uma das ferramentas de inteligência artificial mais promissoras, concebidas para suportar terminais rádio inteligentes. Espera-se que futuros terminais móveis inteligentes habilitados para operar em 5G acessem autonomamente as melhores bandas espectrais com o auxílio de aprendizagem e realizem inferência da eficiência espectral de forma sofisticada, controlando desde a potência de transmissão, ao ajuste da eficiência energética e protocolos de transmissão com o auxílio de aprendizagem e inferência de qualidade de serviço. Nesse sentido, Jiang *et al.* classificam diversos tópicos e trabalhos relacionados ao aprendizado de máquina aplicado à caracterização e análise de espectro em redes sem fio da próxima geração [Jiang et al., 2017]. O trabalho classifica as técnicas em aprendizado supervisionado, não-supervisionado e por reforço. Destaca-se que no contexto de caracterização de espectro diversas técnicas distintas podem ser aplicadas, tendo destaque os algoritmos com aprendizado supervisionado baseado em redes Bayesianas, modelos de regressão, KNN (*K-Nearest Neighbor*) e SVM (*Support Vector Machines*).

A captura de quadros sofre com as limitações da necessidade de uma boa recepção e de sintonização com o canal para captura com sucesso, apesar de haver uma probabilidade pequena de captura de quadros em canais adjacentes. Assim, transmissões que podem interferir com a rede sem fio, como *bluetooth*, fornos de micro-ondas, telefones sem fio, babás eletrônicas e mesmo quadros que colidiram ou que estejam abaixo do limite de sensibilidade dada a relação sinal-ruído, podem impedir que os quadros sejam vistos pela captura. Assim, uma alternativa à captura de quadros é o monitoramento espectral, que captura toda a energia que chega na antena na faixa de espectro sintonizada, independente da fonte. A captura do espectro é feita através de analisadores de espectro, como WiSpy[1]. Essa captura provê uma melhor visão das fontes de interferência no ponto monitorado e permite definir se uma fonte é um transmissor de uma rede sem fio ou de uma rede *bluetooth*, por exemplo. Isso é possível graças às "assinaturas" de cada tecnologia, isto é, o formato do espectro capturado. Uma limitação da captura espectral é não permitir resposta instantânea, uma vez que para identificar a "assinatura" é necessário uma integração no tempo. Além disso, existe um elevado custo financeiro associado à utilização de analisadores de espectro, principalmente em redes de grande porte.

Os analisadores de espectro de baixo custo como o WiSpy são constituídos basicamente por um *chipset* que implementa o IEEE 802.11 com um *firmware* específico capaz de usar o processador de sinais digitais do Wi-Fi na captura de energia em uma pequena faixa do espectro. Seguindo essa mesma ideia, é possível acessar o analisador de espectro no *chipset* Atheros de pontos de acesso que usam a distribuição *OpenWRT*, permitindo

[1]Disponível em https://www.metageek.com/products/wi-spy/

transformar pontos de acesso comuns em analisadores de espectro de baixo custo. O processo de captura utilizando esses pontos de acesso é feito em *hardware*, analisando cada quadro recebido. As medidas são extremamente rápidas e a quantidade de dados coletada é limitada, principalmente pelo tempo de transmissão e armazenamento.

Diversos trabalhos usam a caracterização do espectro em uma área como ferramenta para melhorar o desempenho das redes. Wang *et al.* propõem a otimização do uso do espectro das redes sem fio através da cooperação entre pontos de acesso para realizar formatação de feixe (*beamforming*) [Wang et al., 2018]. Os pontos de acesso são conectados via Ethernet para tornar mais ágil a troca de informações. A proposta baseia-se em três pilares: um esquema cooperativo que permite o sistema estimar desvios de fase em cada símbolo transmitido e dinamicamente ajustar as fases para garantir o alinhamento; um mecanismo de estimação que mede a qualidade do canal; e um algoritmo aleatório de escolha de usuários para realizar formatação de feixe com custo computacional constante. Os autores mostram que o algoritmo aleatório é capaz de escalar a rede linearmente e tem desempenho equivalente a 70% comparado a algoritmos mais complexos.

Caracterização do comportamento de usuários

Os padrões de mobilidade humana refletem muitos aspectos da vida, desde a disseminação global de doenças infecciosas até o planejamento urbano e padrões diários de deslocamento. Nos últimos anos, a prevalência de métodos e tecnologias de posicionamento, tais como o sistema de posicionamento global, geo-posicionamento de torre de rádio celular e sistemas de posicionamento Wi-Fi, têm direcionado esforços para coletar dados de mobilidade humana e extrair padrões de interesse dentro desses dados, com o objetivo de promover o desenvolvimento de serviços e aplicações baseados em localização. Os esforços para extrair padrões significativos em dados de mobilidade em grande escala têm solicitado o uso de técnicas de análise avançadas para mineração de dados, geralmente baseadas em métodos de aprendizado de máquina.

A abordagem para análise de posicionamento baseada em impressão digital (*fingerprints*) é comumente usada para posicionamento *indoor* e consiste de duas fases: uma *online* (treinamento) e outra *offline* (posicionamento). Na fase *offline*, um mapa rádio é criado usando valores de Força de Sinal Recebido (*Received Signal Strength* - RSS) que são medidos nos pontos de acesso existentes no ambiente. Os mapas de rádio incluem, além dos valores RSS, informações dos pontos de acesso em que as medidas foram feitas. Na fase *online* a localização é realizada combinando os valores RSS do mapa de rádio e os valores de RSS medidos pela unidade móvel. É interessante destacar que mudanças físicas de dispositivos entre as fases *offline* e *online* podem afetar a precisão do posicionamento, assim como a escolha de algoritmos e seus parâmetros na fase *online*. Na literatura de posicionamento, os algoritmos de aprendizado de máquina têm amplo uso na estimativa dessas posições. Diversos são os algoritmos sugeridos nessa aplicação, sendo uma tarefa não trivial a descoberta daquele que melhor se comporta em um dado cenário. Na comparação entre os algoritmos com relação ao posicionamento e tempo de computação, Bokzurt *et al.* mostram que o algoritmo k-vizinhos mais próximos (*K-Nearest Neighbors* - k-NN) é o mais adequado [Bozkurt et al., 2015] quando comparado aos algoritmos de Árvore de Decisão, Naïve Bayes, Rede Bayesiana, Otimização Sequencial Mínima (*Sequential Minimal Optimization* - SMO), AdaBoost e Bagging.

A disponibilidade de grandes conjuntos de dados utilizados para o rastreamento da localização do usuário se tornou mais realista a partir da chegada das diversas tecnologias relacionadas às telecomunicações e associadas à realidade do *Mobile Big Data* (MBD). Estudos analisam padrões de mobilidade usando modelos estatísticos que capturam propriedades gerais desses padrões observados. Gonzalez *et al.* analisa 100.000 traços (*traces*) de telefones celulares e identifica que a distância entre os usuários segue uma distribuição de probabilidade de lei de potência. Após a identificação dos padrões principais de movimentação de pessoas e do grau de previsibilidade [Song et al., 2010], abriu-se caminho para a pesquisa e aplicação da análise da mobilidade. Outra área interessante se baseia em teoria da complexidade de redes e ferramentas estatísticas. Esses estudos têm como foco a análise das relações sociais a fim de agregar padrões de mobilidade em grandes massas de dados.

Toch *et al.* classificam aplicações de caracterização de mobilidade de usuários em três categorias: (1) aplicações de modelagem do usuário; (2) aplicações de modelagem da localidade; (3) aplicações de modelagem da trajetória [Toch et al., 2018]. Em (1) é analisado apenas um único usuário por um período de tempo, objetivando prever seu padrão de mobilidade e sua localização futura. Em (2) apenas localidades, ou áreas, são levadas em consideração, visando prever a quantidade de pessoas que devem passar por uma certa área. Já em (3) uma análise espaço-temporal de pontos é criada para identificar padrões de mobilidade do usuário, tentando identificar agrupamento de usuários com perfis semelhantes e sua trajetória futura.

É natural imaginar a MBD associada a essas aplicações, assim como a necessidade de um processamento eficiente em relação ao tempo de resposta. Dessa forma, a abordagem da modelagem é de suma importância. Diferentes abordagens usam diferentes métodos, o que influenciará diretamente no resultado. A maioria dos métodos se baseia em aprendizado de máquina, seja de forma supervisionada ou não. Outros métodos utilizam análise de séries temporais não lineares, modelos Markovianos ou regressão.

4.2.2. Principais ferramentas para coleta de dados em redes sem fio

Abordagens de monitoramento de rede são classificadas em ativas ou passivas. As abordagens ativas, implementadas por ferramentas comuns, como o `ping` e `traceroute`, injetam tráfego em uma rede para executar diferentes tipos de medições. As abordagens passivas, por outro lado, não interferem diretamente no tráfego da rede. As ferramentas que adotam tal abordagem apenas observam o tráfego existente à medida que ele passa por um ponto de medição. Esse método fornece mais informações sobre o tráfego de rede, uma vez que pacotes completos podem ser capturados e analisados [Hofstede et al., 2014]. Exportar apenas informações de fluxos é uma variação de abordagem passiva de monitoração para torná-la mais escalável. Nessa abordagem, pacotes de rede são agregados em fluxos e os dados sobre os fluxos são exportados para armazenamento e análise futura. Um fluxo é definido como um conjunto de pacotes passando por um ponto de observação na rede durante um certo período de tempo, de tal forma que todos os pacotes apresentem um conjunto de propriedades comuns [Hofstede et al., 2014]. Essas propriedades comuns podem incluir campos de cabeçalho de pacote, como origem e destino, endereços IP, números de portas, tipo de protocolo de transporte, entre outros. Neste capítulo, discute-se algumas das ferramentas mais importantes, como SNMP, Net-

Flow, sFlow e IPFIX, apresentando suas principais características, limitações e uso em redes sem fio.

O **SNMP** é um protocolo de gerência da camada de aplicações da pilha de protocolos TCP/IP, que utiliza serviços do protocolo de transporte UDP para enviar e receber suas mensagens através da rede. Resumidamente, o protocolo é utilizado para obter informações de servidores SNMP, através de requisições que um gerente faz a um agente. Os agentes são instalados em determinados dispositivos de rede, sobre os quais existe interesse de monitoramento. Cada elemento de rede tem variáveis que representam suas informações e seu estado. O gerente pode acessar essas informações e, até mesmo, mudar o estado de alguma propriedade do elemento que está sendo monitorado. Nesse sentido, cada elemento de rede gerenciado deve ter um agente SNMP e uma base de informações de gerenciamento (*Management Information Base* - MIB). A MIB contém informação sobre os objetos gerenciados, que são abstrações do mundo real representando recursos do sistema que poderão ser consultados ou alterados. Os objetos gerenciados podem ter permissões para consulta (leitura) e até mesmo de alteração. Nesse sentido, leituras representam o estado atual do recurso do elemento de rede monitorado e alterações refletem no recurso monitorado em si. Por exemplo, pode-se usar SNMP para acessar (leitura) configurações de um ponto de acesso, tais como interfaces ativas, canal em uso e potência.

Atualmente existem três versões principais do protocolo (SNMPv1, SNMPv2 e SNMPv3). O SNMP apresenta algumas limitações bem conhecidas. o protocolo, por exemplo, não é apropriado para o gerenciamento de redes muito grandes. Como ele é baseado em um mecanismo de sondagem (*polling*), gerenciar muitos elementos de rede pode provocar atraso excessivo. Ademais, o SNMP básico não tem mecanismos de autenticação. Por fim, destaca-se que o uso de SNMP gasta recursos computacionais, de forma que aplicá-lo em redes sem fio pode exaurir os recursos dos elementos de rede. Nesse sentido, SNMP não é apropriado para redes de sensores sem fio.

O **NetFlow** é um protocolo de rede proprietário da Cisco usado para análise de fluxos. Ele permite a coleta e agregação de informações sobre o tráfego de rede que entra ou sai de um elemento de rede que tem o protocolo habilitado.Os registros de informações coletados pelo NetFlow são enviados por mensagens do protocolo a um local centralizado na rede. Nesse local, os dados podem ser analisados por um administrador de rede que pode determinar, entre outros, a origem e o destino do tráfego, as classes de serviço de tráfego na rede e causas de possíveis congestionamento na rede. Nesse sentido, exemplos do uso típico das estatísticas coletadas pelo protocolo NetFlow são: (i) monitoramento de banda em enlances; (ii) detecção de ameaças em redes, como ataques de negação de serviço (*Denial of Service* - DoS); (iii) contabilidade de uso e cobrança; (iv) investigação das causas de congestionamento e lentidão dos recursos de rede e aplicações. A Figura 4.2 mostra uma configuração típica de monitoramento usando o NetFlow, que consiste em três componentes principais [Hofstede et al., 2014]. O Exportador NetFlow agrega pacotes em fluxos e exporta registros de fluxos para um ou mais coletores de fluxo. A agregação é feita com base nos endereços IP de origem/destino, portas de origem/destino, classes de serviço, protocolo IP e interface de origem. O Coletor NetFlow é responsável pela recepção, armazenamento e pré-processamento de dados de fluxos recebidos de um exportador de fluxos. Já o Analisador analisa os dados de fluxos recebidos. Essas análises cobrem contextos, como detecção de intrusão e geração de perfis de tráfego.

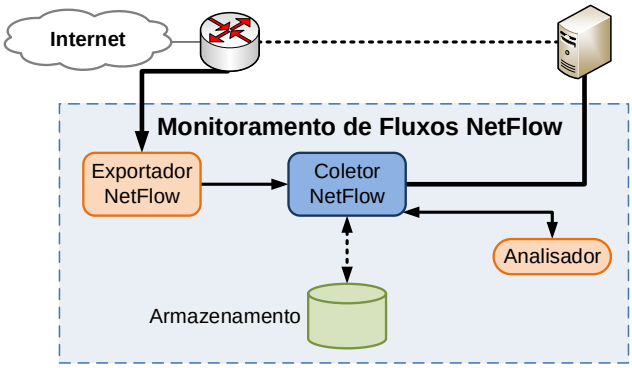


Figura 4.2. Configuração típica de monitoramento de fluxo usando o NetFlow. Os dispositivos habilitados com o protocolo NetFlow, exportadores NetFlow, criam registros, agregando os pacotes em fluxos e exportando os registros para coletores de fluxo. Os coletores NetFlow armazenam e pré-processam os dados recebidos e encaminham para o analisador que consome os dados para gerar informação.

Cada pacote que será encaminhado é examinado e o primeiro pacote que se encaixa a algum parâmetro faz com que uma entrada seja criada na *cache* do NetFlow, criando assim um registro. O pacote é encaminhado e, pacotes que o sucedem e que também se encaixam nos mesmos parâmetros do primeiro, são agregados ao registro de fluxo recém-criado. Os contadores desse registro são atualizados a cada novo pacote que casa com seu padrão. Os fluxos que estão na *cache* são exportados para um coletor de fluxo (*NetFlow collector*) que, por sua vez, armazena os dados do fluxo em uma base regular. O envio para os coletores de fluxos é realizado por exportadores do NetFlow que enviam registros usando UDP ou SCTP, quando o transporte confiável é necessário. Por fim, os fluxos podem ser analisados por aplicações de análise (*Analysis Applications*). Essas aplicações analisam os dados de fluxo recebidos para fins de detecção de intrusão ou perfil de tráfego. Elas são responsáveis pela apresentação dos dados e criação de relatórios.

Um dos principais pontos fracos relacionado ao NetFlow refere-se à sobrecarga que ele pode impor à infraestrutura de rede. A coleta e o envio dos fluxos pode adicionar sobrecarga aos roteadores e comutadores, por muitas vezes já sobrecarregados. Além disso, quanto mais detalhes os administradores tentam inserir em uma única tupla de fluxo, mais sobrecarga é produzida. Em muitos casos, administradores de rede desabilitam o protocolo para não prejudicar o desempenho das redes. Ressalta-se que o NetFlow tem visibilidade limitada do tráfego, ou pacotes, encaminhados. Como consequência, a comunicação interna (LAN/VLAN) não é contabilizada nos fluxos exportados.

Existe um conjunto de ferramentas alternativas ao NetFlow. São variações que seguem a mesma linha do NetFlow disponibilizadas por fabricantes como Juniper (jFlow), Ericsson (rFlow), Huawei (NetStream) e HP (sFlow). A tecnologia **sFlow**[2], por exemplo, foi desenvolvida pela InMon Inc. que se tornou um padrão da indústria definido na RFC 3176 [Li et al., 2013]. A tecnologia utiliza amostragem aleatória e o agente sFlow é incorporado em comutadores e roteadores de diversos fabricantes. O agente sFlow é um processo de software que associa contadores de interface e amostras de fluxo, gerando datagramas sFlow datagramas que são imediatamente enviados para os coletores sFlow

[2]Coletores para sFlow estão disponíveis em https://sflow.org/products/collectors.php

através de datagramas UDP. Os datagramas sFlow contém informações sobre a versão sFlow, endereço do agente, IP de origem, número de sequência, quantidade de amostras e, normalmente, até 10 amostras de fluxo. A solução de envio imediato de dados minimiza o uso de memória e de processamento. Os pacotes são tipicamente amostrados por Circuitos Integrados Específicos de Aplicação (*Application Specific Circuits* - ASICs) para garantir o alto desempenho de velocidade de fio. Os dados do sFlow contêm o cabeçalho completo do pacote e informações de encaminhamento. O sFlow é capaz de executar em camada 2 e, assim, é capaz capturar tráfego que não é IP. Os coletores sFlow são servidores que coletam os datagramas sFlow. Destaca-se também o **IPFIX** (Internet Protocol Flow Information Export), um padrão derivado do NetFlow v9 para exportar as informações sobre os fluxos de rede. O IPFIX também é capaz de exportar qualquer informação de tráfego de L2-L7 para o coletor de fluxo. É um protocolo flexível que suporta campos de comprimento variável. Ele permite coletar informações como URL ou *host*, como `facebook.com`, bem como os tipos de dados definidos pelo usuário.

4.3. Processamento de Grandes Massas de Dados

O processamento de grandes massas de dados é comumente associado a aplicações de Aprendizado de Máquina (*Machine Learning*). O aprendizado de máquina refere-se à capacidade dos computadores de aprenderem sem serem explicitamente programados [Boutaba et al., 2018] e baseia-se na teoria da probabilidade e estatística, teorema de Bayes e otimização, sendo a base para a análise e a ciência de dados (*Data Science*) no cenário de grandes massas de dados (*Big Data*) [Blei e Smyth, 2017]. A transformação dos dados em informação e conhecimento através, principalmente, das técnicas de aprendizado de máquina é a etapa mais importante de qualquer análise de grandes massas de dados [Hu et al., 2014]. A análise dos dados pode ocorrer em lote ou em fluxo. Contudo, previamente à submissão dos dados para análise é necessário prepará-los. Dessa forma, uma etapa essencial antes da análise é o pré-processamento dos dados.

4.3.1. Análise de dados em lotes (*batch*)

O processamento em lotes é a forma mais tradicional de processamento de dados. Utilizam-se base de dados, as quais registram permanentemente as informações que são posteriormente processadas por meio de consultas (*queries*) às bases. Também são formas comuns o uso de Processamento Analítico Online (*Online Analytical Processing* - OLAP), técnicas de mineração de dados [Carbone et al., 2015b] e processamento paralelo com ferramentas como o Hadoop. No processamento da dados tradicional, utilizam-se operações como seleções relacionais, projeções e agregações, entre outras. Os dados são armazenados em disco e são processados em momentos posteriores à sua coleta, não havendo necessidade de resposta em tempo real, como mostrado na Figura 4.3.

O processamento de dados em lotes é uma forma eficiente de processar grandes volumes de dados quando as transações são coletadas em um período espaçado de tempo. Assim, os dados são coletados, armazenados, processados e, por fim, são gerados os resultados do processamento em lote. A arquitetura e implementação das bases de dados tradicionais, utilizadas no processamento em lotes, não foram projetadas para a inserção e leitura rápida e contínua de dados [Andreoni Lopez et al., 2018]. Contudo, isso não gera problemas, pois as técnicas para processamento em lotes não têm requisitos tão fortes

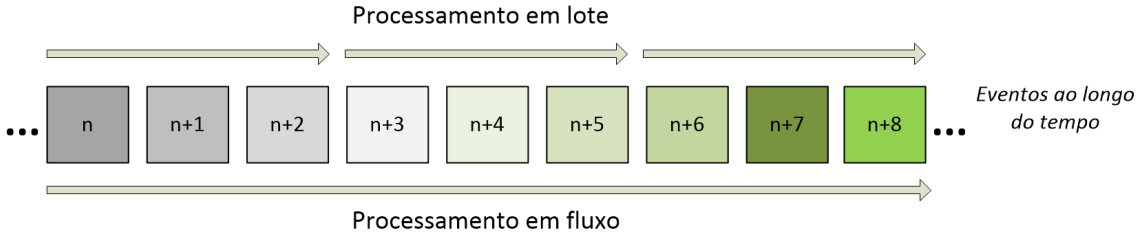


Figura 4.3. Representação do processamento de dados em lote, no qual os dados armazenados em disco são agrupados temporalmente e processados sem necessidade de resposta em tempo real; e em fluxo, no qual os dados em memória são processados assim que recebidos, gerando resposta em tempo real.

com relação à vazão de resultados de processamento. No contexto de controle e monitoramento de redes sem fio, o processamento em lotes é útil, pois dados arquivados são processados gerando informações de interesse que podem ser usadas posteriormente para melhoria da rede.

4.3.2. Análise de dados em fluxo e tempo real

Aplicações que demandam a atuação com baixo tempo de resposta, sejam elas realizadas por um operador ou automaticamente, necessitam combinar a captura de dados em fluxo com o uso de técnicas de inferência e correlação. Devido à alta taxa de chegada de novos dados em fluxo e elevado volume de dados em uma rede de grande porte, os algoritmos de processamento tradicionais carecem de tempo suficiente para percorrer os dados repetidamente de forma iterativa. Para esses cenários, algoritmos de uma única passagem que utilizam pequenas quantidades de memória são necessários, caracterizando a análise de dados em fluxo e tempo real. Essa análise parte do princípio que a massa de dados não é estática e chega constantemente, formando um fluxo de dados, como mostrado na Figura 4.3. Exemplos desse tipo de dados que são produzidos constantemente incluem *logs* de servidores *Web*, *logs* de aplicativos, dados de sensoriamento, mudanças de estado em bases de dados transacionais [Carbone et al., 2015b]. Nesse sentido, o uso do processamento em lotes tradicional torna-se inviável, já que as respostas do processamento só têm validade se respeitarem uma janela de tempo muito curta. Estratégias que fazem o agrupamento de dados por horas, dias ou semanas, para posteriormente processar os dados, geram um atraso intrínseco que pode tornar algumas aplicações específicas inviáveis. Ferramentas para coletas de dados, gerenciadores de fluxo de trabalho e escalonadores funcionam criando um *pipeline* contínuo de lotes [Carbone et al., 2015b].

Embora muitas das informações de interesse de organizações com grandes bases de dados possam ser obtidas com o processamento em lote, algumas aplicações específicas necessitam de um resultado em tempo real da análise dos dados coletados. Isso permite que a organização tome uma atitude imediata em situações que um tempo muito curto pode ser suficiente para causar problemas de diferentes naturezas. Por exemplo, a realização de uma análise para detecção de intrusão demanda uma resposta rápida, para que a intrusão não cause grandes problemas a uma máquina ou à rede. Assim, o principal objetivo do processamento de fluxos em tempo real é conseguir extrair informações para a tomada de decisão em uma janela de tempo muito curta. Ao se realizar uma análise de dados em fluxo e tempo real, os recursos de processamento e memória passam a ser um potencial gargalo, especialmente no contexto de *Big Data*. Os dados chegam em grande

volume, podendo apresentar rajadas, e os resultados devem ser apresentados em tempo real ou quase real. Assim, o uso de memória secundária, em geral, não é possível nesses cenários [Garofalakis et al., 2016], pois o tempo de escrita e de leitura é alto e incompatível com as restrições temporais da aplicação. O processamento em fluxo e tempo real utiliza o esquema "compute primeiro, armazene depois" (*compute-first, store-second*). A necessidade de resposta muito rápida para um grande volume de dados usualmente leva a necessidade do uso de sistemas distribuídos para o processamento da massa de dados.

Outras características também são importantes no processamento em fluxo, quando comparado ao processamento em lote [Andreoni Lopez et al., 2018], tais como, os elementos chegam em linha e devem ser processados tão logo sejam recebidos; o sistema de processamento não tem controle sobre a ordem com que os elementos chegam; os fluxos não têm restrições quanto a tamanho ou duração; e elementos processados são usualmente descartados ou arquivados e, portanto, não são acessados novamente. Os fluxos de dados podem ser representados através de diferentes modelos.

O Modelo de série temporal. Uma série temporal é um conjunto de observações de dados feitas com intervalos de tempo constantes, ao longo de uma janela de tempo. Dada uma observação de uma série temporal, ela pode ser dividida em três componentes básicos, sendo eles: tendência, relacionado a uma visão de longo prazo; sazonalidade, relacionado a movimentos sistemáticos relativos ao calendário; e irregularidade, que são flutuações de curto prazo não sistemáticas. Em uma rede sem fio em uma universidade, por exemplo, a quantidade de usuários em um ponto de acesso costuma sofrer com sazonalidade, pois a quantidade de usuários varia conforme o período de dia e noite, finais de semana e períodos de férias. Usualmente, uma análise de dados realiza um ajuste sazonal, identificando e removendo influências sistemáticas e relacionadas ao calendário. Ainda no exemplo da rede sem fio universitária, os períodos sem usuários devido ao calendário deveriam ser removidos na análise de carga média da rede para dimensionamento e posicionamento dos pontos de acesso. De fato, o processo de ajuste sazonal é importante porque os efeitos sazonais podem mascarar o verdadeiro movimento da série temporal, assim como características não sazonais que sejam de interesse.

O Modelo caixa registradora. Para definir esse modelo, assume-se que o sinal de entrada é representado por um fluxo $a_1, a_2, \ldots$ com chegada sequencial e um sinal A, definido pela função $A[j] : [1 \ldots N] \rightarrow R$. As amostras a_i de entrada são incrementos para o sinal $A[j]$. Por exemplo, seja $a_i = (j, I_i)$, $I_i > 0$ e $A[j]_i = A_{i-1} + I_i$. Nesse caso, A é um fluxo no modelo caixa registradora, em que $A[j]_i$ é o estado do sinal depois da chegada da i-ésima amostra [Lee et al., 2005]. Esse é um modelo de fluxo entre os mais populares, representando, por exemplo, o tempo total de acesso de cada usuário em uma rede após múltiplos acessos, em que j representa cada usuário, I_i é o tempo de cada um de seus acessos e $A[j]_i$ representa o tempo total de acesso do usuário j até o momento i. Outro exemplo constitui o monitoramento de IPs que acessam um ponto de acesso, em que cada IP recebe um valor de j diferente e $A[j]_i$ representa o total de acessos daquele IP.

O Modelo de catraca. Esse modelo é mais genérico que os anteriores, mas é semelhante à caixa registradora. Contudo, no modelo de catraca, I_i pode assumir valores positivos ou negativos. Esse é o modelo mais apropriado para estudar situações amplamente dinâmicas, em que elementos podem ser inseridos ou retirados [Lee et al., 2005].

4.3.3. Técnicas de pré-processamento de dados

Os dados brutos coletados a partir de inúmeras fontes de dados heterogêneas podem ser compostos por uma grande quantidade de dados sem valor ou desnecessários. Isso resulta em conjuntos de dados com níveis de qualidade diferentes, que variam de acordo com a sua completude e intensidade da presença de redundância, ruído e inconsistências [Hu et al., 2014]. Os dados em sua forma bruta, também conhecidos como dados primários [Tsai et al., 2015], apresentam baixa qualidade e ao serem diretamente processados resultam na utilização de um grande espaço de armazenamento e na baixa qualidade da informação gerada. Assim, a qualidade dos dados tem impacto direto no desempenho dos algoritmos de processamento empregados e na qualidade dos resultados obtidos. Dessa forma, uma das primeiras etapas à qual os dados coletados são submetidos antes de serem armazenados para análise posterior é o pré-processamento. O objetivo do pré-processamento é justamente aumentar a qualidade do conjunto de dados obtido e, possivelmente, reduzir o seu volume através da limpeza de inconsistências e ruídos, eliminação da redundância e integração dos dados para fornecer uma visão unificada. Essa etapa pode ocorrer antes ou após a transmissão dos dados, sendo mais adequada a sua execução antes da transmissão, para fins de economia de banda e de espaço de armazenamento [Hu et al., 2014].

Após a realização do pré-processamento, o conjunto de dados obtido pode ser considerado confiável e adequado para a aplicação de algoritmos de mineração de dados [Tsai et al., 2015]. O pré-processamento de dados é de especial importância para as redes sem fio de grande porte devido ao grande volume de dados coletados e à redundância existente neles. Por exemplo, dois pontos de acesso distintos podem coletar dados sobre a rede em uma região onde existe sobreposição de cobertura. Os dados coletados por ambos os pontos de acesso referentes a essa região em comum são fortemente redundantes, sendo desnecessário armazená-los por completo. Além disso, caso fossem processados com a presença da redundância, as conclusões sobre a análise poderiam ser tendenciosas. O pré-processamento inclui diversas técnicas para promover a *limpeza*, a *integração*, a *redução* e a *transformação* dos dados [Hu et al., 2014, García et al., 2016].

Limpeza dos dados. As técnicas para limpeza dos dados determinam a acurácia e a completude dos dados, consertando os problemas encontrados através da remoção ou adição de dados ao conjunto original, e resolvendo inconsistências. Conjuntos de dados reais frequentemente são incompletos devido a falhas no processo de coleta dos dados, limitações no processo de aquisição dos dados ou restrições de custo, que impedem o armazenamento de alguns valores que deveriam estar presentes no conjunto de dados [García et al., 2016]. Além disso, é necessário tratar os erros nos dados, documentando tanto as ocorrências quanto os tipos de erros para que os procedimentos possam ser modificados a fim de reduzir erros futuros [Hu et al., 2014].

Na primeira etapa da limpeza busca-se por dados discrepantes usando diversas ferramentas comerciais de depuração de dados (*data scrubbing tools*) e de auditoria de dados (*data auditing tools*). Ferramentas de depuração de dados usam conhecimentos simples sobre o domínio dos dados para detectar erros e corrigí-los, empregando técnicas de análise (*parsing*) e de correspondência difusa (*fuzzy matching*). Já as ferramentas de auditoria de dados descobrem regras e relações, identificando dados que violam as condi-

ções encontradas. Para tanto, empregam, por exemplo, análise estatística para identificar correlações e algoritmos de agrupamento para identificar *outliers*.

Quando se identifica a ausência de dados, a solução trivial é descartar a instância, mas isso pode resultar na deturpação do processo de aprendizagem e no descarte de informações importantes. Outra abordagem é completar os dados manualmente, o que pode ser impraticável. Uma constante global também pode ser utilizada para completar os dados ausentes, resultando, potencialmente, em uma interpretação errada dos dados. Em vez de uma constante, um valor de tendência central para a característica como a média ou mediana pode ser utilizado. A estratégia mais utilizada é determinar o valor ausente através de modelos probabilísticos aproximados, utilizando procedimentos de máxima verossimilhança [García et al., 2016].

A presença de erros, ou ruído, nos dados é tratada através de duas abordagens principais, polimento de dados (*data polishing*) e filtros de ruído [Liebchen et al., 2007]. O resultado de ambas é semelhante, produzindo uma saída suavizada através de técnicas de regressão, análise de *outliers* e técnicas de categorização (*binning*). Uma sobrecarga computacional extra significativa pode ser adicionada dependendo da complexidade do modelo utilizado. Dessa forma, é necessário buscar o equilíbrio entre o custo adicional da limpeza dos dados e a melhoria da acurácia dos resultados obtidos [Hu et al., 2014].

Integração dos dados. As técnicas de integração de dados, ou resumo, combinam os dados provenientes de diferentes fontes, unificando a visão sobre eles e reduzindo a redundância. A eliminação de redundância também é feita através de técnicas de detecção de redundância e de compressão de dados, permitindo diminuir a sobrecarga na transmissão e no armazenamento dos dados. A redundância pode se apresentar na forma espacial, temporal, estatística ou perceptiva e está fortemente presente em dados de vídeo e imagem. Técnicas para tratamento de redundância nesses casos já são muito bem conhecidas, como JPEG e PNG para imagem e MPEG-2, MPEG-4, H.263 e H.264/AVC para vídeo, mas não são aplicáveis ao contexto de análise de dados em redes sem fio de grande porte. Nesse caso, técnicas de remoção de duplicatas são mais adequadas.

Cada instância pode ser composta por diversas características (atributos) provenientes de fontes diferentes e que podem apresentar esquemas diferentes. Os metadados das características passam a ser importantes para evitar erros durante a integração dos esquemas. Características que podem ser derivadas de outras características ou de um conjunto de características devem ser eliminadas se puderem ser consideradas redundantes. A redundância pode ser avaliada através da análise de correlação, que utiliza o teste *qui-quadrado* (χ^2) para características nominais, e o coeficiente de correlação e covariância para características numéricas. A duplicação de dados também deve ser observada na etapa de integração e a consistência dos dados deve ser garantida, de forma que durante a integração dos dados é necessário detectar e resolver conflitos nos valores das características [García et al., 2016].

Duas estratégias para integração dos dados incluem o uso de Armazém de Dados (*Data Warehouse*) e Federação de Dados (*Data Federation*) [Hu et al., 2014]. No primeiro caso, a integração passa pelas etapas de extração, transformação e carregamento dos dados. Na extração seleciona-se os dados necessários para a análise, que são modificados na etapa de transformação através da aplicação de um conjunto de regras, convertendo

os dados em um formato padrão. Por fim, os dados transformados são importados para a infraestrutura de armazenamento e geralmente são provenientes de uma única fonte [Hu et al., 2014]. No segundo caso, cria-se um banco de dados virtual para consultar e agregar dados de fontes diferentes. O banco de dados virtual contém apenas informação ou metadados sobre os dados reais e sua localização, não armazenando os dados de fato. Essas abordagens são adequadas para processamento de dados em lote, mas ineficientes para dados em fluxo.

Redução dos dados. As técnicas de redução dos dados objetivam diminuir o volume do conjunto de dados, mas de forma que o resultado produzido após o processamento seja mantido. As técnicas incluem estratégias de redução de dimensionalidade e de numerosidade. Também são utilizadas técnicas de compressão de dados para reduzir o volume de dados. A redução de dimensionalidade busca uma representação comprimida dos dados originais, focando na diminuição da quantidade de variáveis aleatórias ou características consideradas. Para tanto são usadas as Transformadas *Wavelet*, a Análise de Componentes Principais (*Principal Component Analysis* - PCA) e Seleção de Características. Também podem ser empregados algoritmos genéticos para reduzir de forma ótima a dimensão do conjunto de dados [Tannahill e Jamshidi, 2014]. Na redução de numerosidade, os dados são substituídos por representações alternativas, de formato menor, usando técnicas paramétricas ou não-paramétricas. As paramétricas, como regressão e modelos log-lineares, estimam os dados e armazenam apenas os parâmetros do modelo que os descrevem. As técnicas não-paramétricas incluem histogramas, algoritmos de agrupamento, técnicas de amostragem e agregação de cubos de dados. Dentre as técnicas de redução, vale destacar as transformadas *wavelet*, o método PCA e a seleção de características.

A *Transformada Wavelet* é uma técnica de processamento de sinais linear em que o sinal passa a ser representado por uma soma de formas de onda mais simples. A transformada *wavelet* tem como objetivo capturar tendências em funções numéricas, decompondo o sinal em um conjunto de coeficientes. Assim, o vetor de dados passa a ser representado por um vetor de coeficientes *wavelet* de mesmo comprimento. Existe uma família de transformadas Wavelet que podem ser usadas no pré-processamento dos dados, sendo as mais populares Haar-2, Daubechies-4 e Daubechcies-6 [Han et al., 2011]. Os coeficientes mais baixos podem ser descartados sem prejuízo significativo para a recuperação dos dados originais. Através do armazenamento de uma pequena fração dos coeficientes de maior intensidade, os dados originais são obtidos aplicando a transformada inversa.

O *PCA* cria um conjunto menor de variáveis que permite representar os dados originais. O PCA é considerado um método de extração de características. Esses dados podem ser descritos através de um vetor de n características ou dimensões e o PCA combina a essência das características originais usando k vetores ortogonais de dimensão n, com $k \leq n$, que melhor representem os dados a serem reduzidos. Como o espaço dimensional de projeção dos dados é menor, a dimensionalidade dos dados é reduzida. Nesse novo espaço dimensional, o PCA pode revelar relações que anteriormente não estavam claras, permitindo interpretações que normalmente seriam ignoradas.

A *Seleção de Características* remove características redundantes ou irrelevantes, mantendo um conjunto mínimo de características que resulte em uma distribuição de probabilidade próxima da distribuição original. A seleção de características reduz o risco de

overfitting dos algoritmos de mineração de dados e reduz o espaço de busca, tornando o processo de aprendizagem mais rápido e consumindo menos memória [García et al., 2016]. Dados representados por um vetor de n características possuem 2^n possíveis subconjuntos, dos quais deve-se escolher uma quantidade ótima para representar os dados. Devido ao grande número de possibilidades que podem ser exploradas para chegar ao resultado ótimo, geralmente são utilizadas heurísticas que exploram um conjunto reduzido do espaço de busca. Geralmente são usados algoritmos gulosos (*greedy*), que escolhem sempre soluções ótimas locais, como seleção passo a passo para frente (*stepwise forward selection*), eliminação passo a passo para trás (*stepwise backward elimination*), uma combinação dos dois anteriores e indução de árvores de decisão. Outra abordagem utiliza o teste de independência qui-quadrado para decidir quais características devem ser selecionadas.

Transformação dos dados: A transformação uniformiza a representação dos dados e geralmente utiliza métodos para reduzir a complexidade e a dimensionalidade do conjunto de dados [Tsai et al., 2015], podendo ser incluída também na etapa de redução dos dados [García et al., 2016]. As principais estratégias para transformar os dados são as técnicas de suavização, construção de características, agregação, normalização, discretização e geração de hierarquias de conceitos para dados nominais. A suavização remove ruído dos dados utilizando, por exemplo, técnicas de regressão e de agrupamento. A construção de características tem como objetivo criar novas características baseadas em outras para melhorar a acurácia e a compreensão de dados de grande dimensionalidade. Juntamente com a seleção de características, a construção de características integra uma área de pesquisa em crescimento conhecida como engenharia de atributos, ou de atributos (*feature engineering*). Os procedimentos de seleção de características identificam e removem o máximo de informação redundante e irrelevante possível [García et al., 2016], enquanto a extração e a construção combinam as características do conjunto original para obter um novo conjunto de variáveis menos redundantes. Na agregação os dados são resumidos, resultando em uma nova representação. O resumo pode ser feito na forma de média, variância, máximo e mínimo. Por exemplo, o consumo diário de energia pode ser agregado para mostrar o consumo mensal máximo, ou anual médio. A normalização modifica a escala dos dados para se ajustarem em uma faixa de valores menor. A discretização substitui a representação de valores numéricos por intervalos numéricos ou rótulos conceituais. Por fim, a geração de hierarquias de conceitos para dados nominais cria múltiplos níveis de granularidade em que os dados podem ser encaixados.

Apesar de a área de pré-processamento de dados já ser bastante explorada, ainda existe muita atividade de pesquisa para buscar novos métodos e aprimorar os métodos existentes, de forma que sejam capazes de lidar com o volume de dados disponíveis cada vez maior e com a geração de conhecimento em tempo real. Existe na literatura trabalhos que estudam novos métodos para, por exemplo, dividir os dados entre processadores em sistemas em nuvem [Ham e Lee, 2014] e realizar seleção de características em dados em fluxo [Andreoni Lopez et al., 2019].

4.3.4. Aprendizado de Máquina

Após a preparação dos dados pelo pré-processamento, a análise de grandes massas de dados transforma os dados em conhecimento usando técnicas de aprendizado de

máquina. O aprendizado de máquina permite inferir soluções para problemas que possam ser representados por um amplo conjunto de dados. As aplicações de aprendizado de máquina são projetadas para identificar e explorar padrões ocultos em dados, para descrever o resultado como um agrupamento de dados em problemas de agrupamento, para prever o resultado de eventos futuros em problemas de classificação e problemas de regressão, e para avaliar o resultado de uma sequência de amostra de dados para problemas de extração de regras [Boutaba et al., 2018]. Assim, de forma simplificada, é possível dividir os problemas de aprendizado de máquina em quatro categorias, agrupamentos (*clustering*), classificação, regressão e extração de regras.

Problemas de agrupamento têm como objetivo minimizar a distância entre amostras de dados com características similares em um mesmo grupo, enquanto maximizam a separação entre grupos distintos. *Problemas de classificação* caracterizam-se por mapearem as entradas em classes-alvos de saída que são representadas pelo conjunto discreto de valores de saída. A coesão dos resultados dos classificadores e dos agrupamentos utiliza frequentemente como parâmetro a Soma dos Erros Quadráticos (*Sum of Squared Errors* - SSE), Equação 1, que leva em consideração o número de agrupamentos, k; a quantidade de dados no i-ésimo agrupamento, n_i; o j-ésimo dado no i-ésimo agrupamento, x_{ij}; a média dos dados no i-ésimo agrupamento, c_i; e a quantidade de dados, n [Tsai et al., 2015]. A Soma dos Erros Quadráticos é dada por

$$SSE = \sum_{i=1}^{k} \sum_{j=1}^{n_i} D(x_{ij} - c_i) \qquad (1) \qquad\qquad c_i = \frac{1}{n_i} \sum_{j=1}^{n_i} x_{ij}. \qquad (2)$$

A métrica de distância, D, mais comum é a distância Euclidiana, definida na Equação 3, onde p_i e p_j são posições de dois dados diferentes. Outras distâncias podem ser usadas, como as distâncias Manhattan e Mikowski.

$$D(P-i, p_j) = (\sum_{l=1}^{d} |p_{il}, p_{jl}|^2)^{\frac{1}{2}} \qquad (3)$$

Métricas usadas para avaliar os resultados da classificação são a acurácia (Equação 4), a precisão (Equação 5), a revocação (Equação 6), a especificidade 7 e F-measure (Equação 8), que permitem descobrir os acertos (verdadeiros positivos, VP, e verdadeiros negativos, VN) e as falhas na classificação, como a quantidade de dados que não pertencem ao grupo mas são incorretamente classificadas como pertencentes (falsos positivos, FP) e a quantidade de dados que pertencem ao grupo que não são classificados como pertencente ao grupo (falsos negativos, FN). Essa interpretação pode ser sumarizada através de uma matriz de confusão do classificador, conforme indicado na Tabela 4.1. Outra métrica importante na avaliação de classificadores é a Área Abaixo da Curva ROC (*Receiver Operating Characteristic* - ROC), conhecida como AUC (*Area Under the ROC Curve*), que permite verificar o compromisso entre a taxa de verdadeiros positivos e a taxa de falsos positivos. O tamanho da AUC é diretamente proporcional ao desempenho do classificador. Andreoni Lopez et al. [Andreoni Lopez et al., 2018] explicam detalhadamente cada uma dessas métricas.

	Classificação positiva	Classificação negativa
Positivo (P)	Verdadeiro positivo (VP)	Falso negativo (FN)
Negativo (N)	Falso positivo (FP)	Verdadeiro negativo (VN)

Tabela 4.1. Matriz de confusão

$$acc = \frac{VP+VN}{P+N}. \quad (4) \qquad p = \frac{VP}{VP+FP}. \quad (5) \qquad r = \frac{VP}{VP+FN}. \quad (6)$$

$$e = \frac{VN}{FP+VN}. \quad (7) \qquad F = \frac{2pr}{p+r}. \quad (8)$$

Os *problemas de regressão* tendem a gerar modelos matemáticos que tendem a se comportar como funções multivariáveis em que os valores de entrada são mapeados em valores contínuos de saída. Já os *problemas de extração* de regras são diferentes dos demais, pois o objetivo desse problema não é inferir valores de saída para cada conjunto de entrada, mas identificar relações estatísticas entre os dados.

Os principais paradigmas de aprendizagem de máquina são supervisionados, não-supervisionados, semi-supervisionados e aprendizado por reforço (*reinforcement learning*). A definição do paradigma de aprendizagem a ser usado em uma aplicação de aprendizado de máquina determina como os dados devem ser coletados, o estabelecimento da verdade básica responsável pela inserção de rótulos dos dados e a engenharia de características responsável por estabelecer as características dos dados e quais delas devem ser exploradas na aplicação. A Figura 4.4 mostra o processamento em fluxo e em lote com processos que representam os quatro paradigmas de aprendizado de máquina. O *aprendizado supervisionado* baseia-se em um conjunto de dados rotulados, chamado de conjunto de treinamento, para criar o modelo de classificação ou regressão dos dados. Esse paradigma de aprendizagem requer a existência *a priori* de um conjunto de

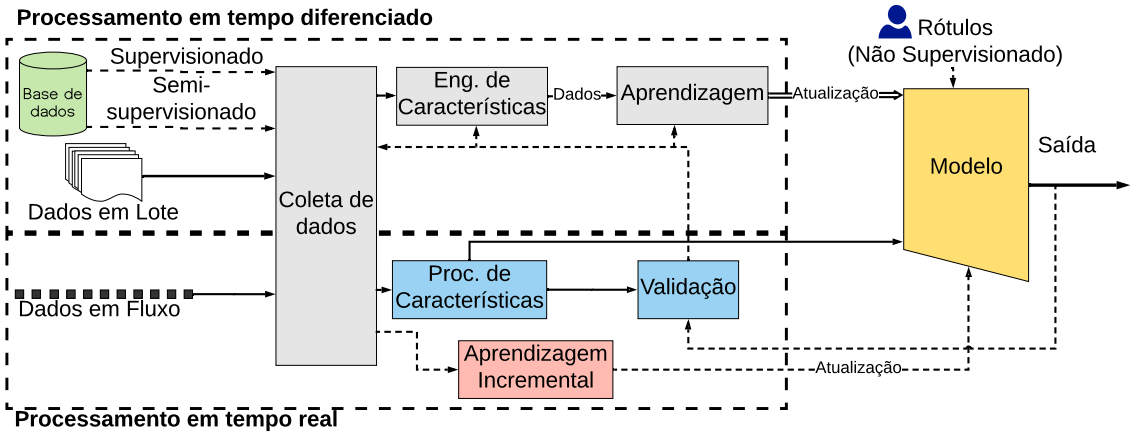


Figura 4.4. Representação dos quatro paradigmas de aprendizado de máquina em um sistema de processamento de dados em lote e em fluxo. A característica contínua e ilimitada dos dados viabiliza o aprendizado incremental, em que o modelo aprende com os dados entrantes. Já o processamento em lote, o treinamento do modelo é somente realizado com bases históricas armazenadas.

dados rotulados para a criação do modelo. Por sua vez, o paradigma de *aprendizado semi-supervisionado* suporta o uso de conjuntos de treinamento com rótulos faltantes ou incompletos. Já o *aprendizado não-supervisionado* busca padrões nos dados de treinamento e, portanto, não requer a existência prévia de dados rotulados. O aprendizado não-supervisionado é adequado para problemas de agrupamentos (*clustering*), em que se busca um padrão de agrupamento dos dados para maximizar a distância entre grupos, enquanto se minimiza a distância entre dados dentro de um mesmo grupo. Por fim, o paradigma de *aprendizado por reforço* consiste em um processo iterativo, em que agentes aprendem efetivamente novos conhecimentos na ausência de qualquer modelo explícito do sistema, com pouco ou nenhum conhecimento do ambiente [Tesauro, 2007]. A ideia central do aprendizado por reforço é que o aprendizado de um agente é baseado em exemplos do conjunto de treinamento, que interage com o mundo externo e aprende com reforços providos pelo ambiente. Os reforços providos podem ser recompensas ou penalidades. Assim, o conjunto de treinamento consiste de pares de amostras de dados e reforços, sejam recompensas ou penalidades. A retroalimentação do ambiente impulsiona o agente para a melhor sequência de ações. O aprendizado por reforço é adequado para problemas de tomada de decisão, planejamento e programação [Tesauro, 2007].

Estabelecer a verdade básica consiste na descrição formal, rótulos, dada às classes de interesse. Os métodos para rotular conjuntos de dados usando as características de uma classe são variados. O método mais elementar é a rotulagem manual por especialistas, com auxílio de ferramentas específicas que, por exemplo, realizam a inspeção profunda de pacotes (*Deep Packet Inspection* – DPI) [Andreoni Lopez et al., 2019] , correspondência de padrões como em sistemas de detecção de intrusão ou técnicas não-supervisionadas como o agrupamento de dados brutos [Zhang et al., 2016]. Uma alternativa a modelos manuais é o desenvolvimento de modelos estatísticos e estruturais que descrevem os dados para inferir uma classe de interesse. Contudo, a definição da verdade básica no estabelecimento do conjunto de treinamento está estreitamente relacionada à acurácia e à precisão do modelo de aprendizado de máquina. A dependência mútua inerente do tamanho dos dados de treinamento de uma classe de interesse em relação às outras impacta o desempenho do modelo [Andreoni Lopez et al., 2018]. Muitas técnicas de aprendizado de máquina assumem que as classes de interesse têm distribuição semelhante no conjunto de treinamento.

O aprendizado de máquina é uma ferramenta poderosa para obter conhecimento a partir de dados coletados em um determinado cenário. Dados provenientes de redes sem fio podem trazer diversas informações escondidas. Especificamente para redes Wi-Fi, através da coleta e análise dos dados é possível descobrir perfis de uso da rede [Afanasyev et al., 2010, Fan et al., 2016], determinar a localização dos usuários [Huang et al., 2016] e monitorar o deslocamento [Chilipirea et al., 2016], permitindo inclusive descobrir padrões de comportamento na movimentação dos usuários [Zeng et al., 2015], reconhecimento de atividades [Alsheikh et al., 2016] e identificação do próprio usuário [Zeng et al., 2016].

4.4. Ferramentas para Processamento de Grandes Massas de Dados

Em redes sem fio, o modelo ideal para processamento de grandes massa de dados é o processamento em fluxo, devido à característica dinâmica e ao volume de dados não limitado gerado pelas redes sem fio. Assim, são originadas diversas oportunidades para

a indústria e a pesquisa gerarem conhecimento e inteligência. No entanto, esse grande volume de dados também traz diversos desafios computacionais, já que a capacidade de processamento de uma única máquina não acompanhou o crescimento do volume de dados. Com isso, surge a necessidade de um sistema distribuído de processamento [Zaharia et al., 2012b]. Nesse contexto, inúmeros modelos de programação de aglomerados computacionais (*clusters*) surgiram visando o tratamento de diversas cargas de trabalho [Low et al., 2012, Isard et al., 2007, Dean e Ghemawat, 2008].

O processamento em fluxo (*stream processing*) possibilita a análise de dados em tempo real, como a análise de arquivos de *logs* de servidores em tempo real ou análise de fluxos de rede a medida que os pacotes chegam aos dispositivos. Ferramentas de processamento de dados em lote são utilizadas para trabalhar com grandes massas de dados estáticos, a ferramenta mais utilizada de processamento de dados em lote é o Hadoop [Dean e Ghemawat, 2008]. Para análise de dados em redes é necessária uma ferramenta de processamento de dados em fluxo, pois os fluxos de rede são dinâmicos. Existem diversas propostas de plataformas para processamento em fluxo como Apache Spark Streaming [Zaharia et al., 2012b], Apache Storm [Iqbal e Soomro, 2015] e Apache Flink [Carbone et al., 2015a]. O Apache Storm é bastante utilizado, mas o Flink possui um desempenho melhor quanto ao processamento em fluxo e o Spark possui um sistema de recuperação de falhas superior [Andreoni Lopez et al., 2016, Chintapalli et al., 2016]. O Apache Hadoop, adotado no processamento em lote e usado como base para outras ferramentas como o Apache Spark, e o Apache Storm são explorados em trabalhos anteriores [Andreoni Lopez et al., 2018].

4.4.1. Apache Spark

Uma ferramenta de processamento distribuído de dados em fluxo amplamente utilizada no mercado é o Apache Spark Streaming. O Spark é uma ferramenta de processamento distribuído em lote que possui extensões que dão suporte a diversas cargas de trabalho. O Spark possui extensões para SQL, aprendizado de máquina, processamento de gráficos e processamento em fluxo, utilizando o conceito de microlotes (*microbatch*) [Zaharia et al., 2012b]. Possui também os módulos YARN e Mesos para a gerência dos aglomerados computacionais (*clusters*), permitindo que o usuário foque na programação de novos aplicativos sem se preocupar com a localização dos dados ou em qual nó o dado será processado. A generalidade de propósito do Spark contribui para diversos benefícios: (i) facilidade no desenvolvimento de aplicações, pois o Spark possui uma API unificada; (ii) maior eficiência na execução de tarefas de processamento e no armazenamento em memória, pois o Spark pode executar diversas funções sobre os mesmos dados, geralmente na memória, enquanto outros sistemas exigem a gravação de dados em armazenamento não volátil para que outros mecanismos os acessem; (iii) possibilidade de criação de novas aplicações, como *queries* interativas em gráficos e aprendizado de máquina em linha (*online*), o que não é possível em outras ferramentas.

Para obter resiliência no armazenamento dos dados, o Spark pode utilizar o HDFS ou o serviço simples de armazenamento (*Simple Storage Service* - S3), que são sistemas de arquivos para dividir, espalhar, replicar e gerenciar dados nos nós de um aglomerado computacional [Andreoni Lopez et al., 2018]. Com isso, os dados armazenados em disco serão distribuídos em diversos nós do aglomerado computacional, assim como os dados

processados em memória, criando um ecossistema tolerante a falhas. O compartilhamento de dados é a principal diferença entre o Spark e outras plataformas do paradigma *MapReduce*, tornando o Spark mais rápido que plataformas anteriores para consultas interativas e algoritmos iterativos, como aprendizado de máquina [Xin et al., 2013]. O Spark possui um paradigma de programação similar ao *MapReduce* do Hadoop, porém possui sua própria abstração de compartilhamento de dados, o Conjunto de Dados Distribuídos Resilientes *(Resilient Distributed Dataset* - RDD). O RDD é a principal estrutura de dados do Spark e é responsável por armazenar os dados em memória e distribuí-los garantindo a resiliência [Andreoni Lopez et al., 2018]. Para cada transformação de dados, como `map`, `filter` e `groupBy` é criado um novo RDD. A manipulação dos RDDs é feita por meio de uma API que pode ser utilizada pelas linguagens Scala, Java, Python e R, em que os usuários podem simplesmente passar funções locais para serem executadas no aglomerado computacional.

O Código 4.1, escrito em Python, demonstra a criação de um RDD que lê mensagens de erro dentro de um arquivo de *log* armazenado em HDFS. O Spark percorre todas as linhas do arquivo em tempo real, mesmo que o servidor esteja gravando novos dados no fim do arquivo, e cria outro RDD somente com as linhas que contêm a palavra inicial "*ERROR*". A 5ª linha define um RDD criado a partir do arquivo de *log* armazenado no HDFS, mediante a coleta de linhas no arquivo. A 6ª linha chama a transformação `filter` para derivar um novo RDD a partir do RDD *log*. A última linha chama a função `count`, outro tipo de operação sobre RDD chamada de ação (*action*), que retorna um resultado, não outro RDD, nesse caso, o número de elementos em um RDD.

```
1 from pyspark import SparkContext, SparkConf
2
3 conf = SparkConf().setAppName("ErrorCount").setMaster("local[*]")
4 sc = SparkContext(conf=conf)
5 log = sc.textFile("hdfs://...")
6 erros = lines.filter(lambda x: x.startswith('ERROR'))
7 print('Total de erros: s%' % errors.count())
```

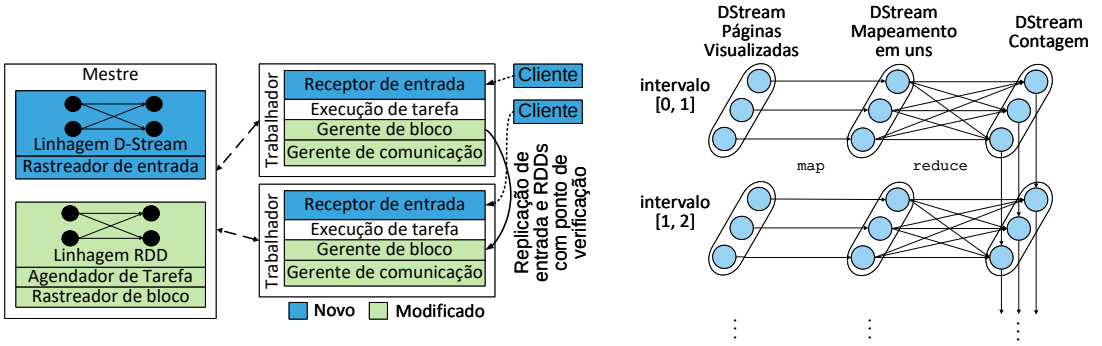
Listing 4.1. Código de leitura de Log utilizando a API pySpark

Sistemas de computação distribuída costumam prover tolerância a falha através de replicação de dados ou criando pontos de verificação (*checkpoints*). Já o Spark utiliza uma abordagem diferente chamada linhagem (*lineage*). Cada RDD grava um grafo de todas as transformações que foram utilizadas para construí-lo e retorna essas operações para uma base de dados para uma futura reconstrução no caso de perda de dados. Recuperação baseada em linhagem é significantemente mais eficiente do que replicação, obtendo um ganho de tempo, pois transferir dados pela rede é muito mais lento do que a escrita na memória RAM e no armazenamento. A recuperação é muito mais rápida do que simplesmente executar novamente o programa, pois o nó com falha geralmente possui múltiplas partições RDDs e essas partições podem ser reconstruídas em paralelo com outros nós [Zaharia et al., 2012a]. O Apache Spark possui quatro bibliotecas de alto nível que fazem o uso da estrutura RDD para uma variedade de funções. A **Spark SQL** implementa *queries* em linguagem de consulta estruturada (*Structured Query Language* - SQL) em banco de dados relacional. A ideia principal dessa biblioteca é ter o mesmo *layout* de banco de dados analíticos dentro dos RDDs. O Spark SQL provê uma abstração

de alto nível para transformações de dados chamadas de *DataFrames*, que são abstrações comuns para dados tubulares em Python e R, com métodos programáticos para filtros e agregações [Armbrust et al., 2015]. A **GraphX** provê uma interface de computação de operações sobre grafos similar ao Pregel [Malewicz et al., 2010] e GraphLab [Low et al., 2014] e implementa a otimização de posicionamento desses sistemas por meio de sua opção de função de particionamento para o RDD [Gonzalez et al., 2014]. A **MLlib** é a biblioteca responsável pelas funções de aprendizado de máquina, possuindo algoritmos de classificação, regressão, agrupamentos, filtros colaborativos, engenharia de características, construção de *pipelines*, e outras funções. O MLlib permite que algoritmos de aprendizado de máquina operem de forma distribuída em aglomerados de computadores. O MLlib foi escrito em Scala e usa bibliotecas nativas de álgebra linear baseadas em C++. MLlib possui APIs para Java, Scala e Python e é distribuído como parte do projeto Spark [Meng et al., 2016]. MLlib possui uma documentação rica, descrevendo todos os métodos suportados e códigos de exemplo para cada linguagem suportada.

A quarta biblioteca de alto nível do Spark o habilita para o processamento de dados em fluxo. O **Spark Streaming** implementa um modelo de processamento de fluxo discretizado (*Discretized Streaming* - D-Streams), que viabiliza a tolerância a falhas e trata os dados retardatários introduzidos por nós mais lentos [Zaharia et al., 2013]. O Spark Streaming é composto por três componentes. O **Mestre** que monitora o grafo de linhagem do D-Stream e organiza as amostras para calcular novas partições RDD. Os **Nós trabalhadores**, que recebem e processam os dados, também armazenam as partições de entrada e calculam os RDDs. O **Cliente** que envia os dados para a ferramenta. O Spark Streaming modifica e adiciona vários componentes ao Spark para ativar o processamento em fluxo. A Figura 4.5(a) mostra os componentes novos e modificados pelo Spark Streaming. A comunicação de rede, por exemplo, é reescrita para permitir que tarefas com entradas remotas sejam recuperadas rapidamente. Existem modificações no mecanismo de linhagem, como remoção de grafos de linhagens que já possuem ponto de verificação (*checkpoint*), assim evitando que os grafos cresçam indefinidamente. Do ponto de vista arquitetural, o que diferencia o Spark Streaming de outras ferramentas que operam sobre dados em fluxo é a divisão das tarefas computacionais em instâncias determinísticas, curtas, sem estado, cada uma delas podendo ser executada em qualquer nó do aglomerado computacional, ou até mesmo em vários nós. O D-Stream discretiza um fluxo de dados em pequenos lotes chamados de microlotes para que eles possam ser tratados pelas tarefas computacionais adequadas.

As aplicações de *big data* costumam utilizar duas abordagens para lidar com tolerância a falha: replicação e *upstream backup*. Em replicação há duas cópias dos dados processados e registros de entrada são replicados para outros nós, porém somente a replicação não é suficiente. É necessário também um protocolo de sincronização para garantir que as cópias de cada operador vejam os dados herdados na mesma ordem. Replicação é custosa computacionalmente, embora a recuperação de uma falha seja rápida. Já em *upstream backup* cada nó retem uma cópia do dado enviado desde um determinado ponto de verificação. Quando um nó falha, uma máquina em modo de espera assume o seu papel e os nós pais reproduzem as mensagens a essa máquina, que antes estava em modo de espera, para que ela mude o seu estado. Ambas as abordagens possuem desvantagens. Replicação tem um grande consumo de recursos de hardware e *upstream*



(a) Arquitetura do Spark Streaming, mostrando seus componentes. Adaptado de [Zaharia et al., 2013]

(b) Grafo de linhagem de um programa de contagem de visualização de páginas WEB. Adaptado de [Zaharia et al., 2013]

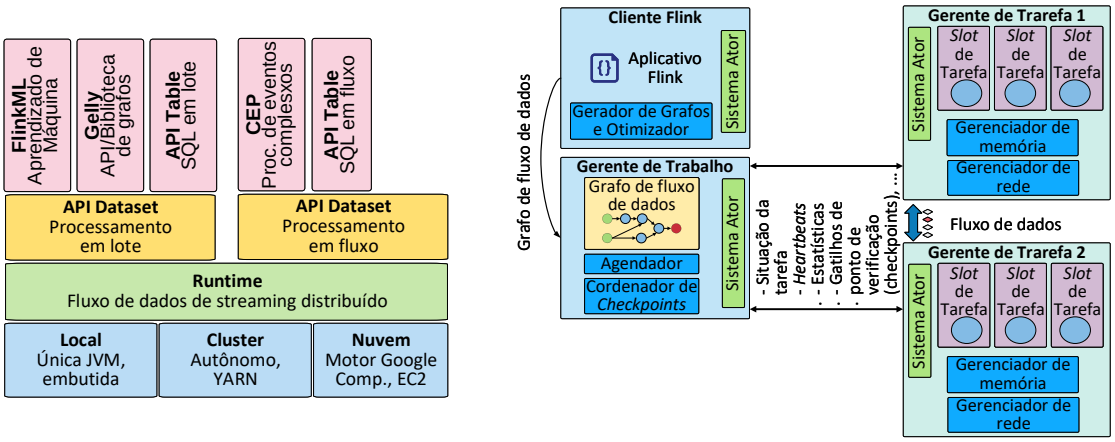
Figura 4.5. Arquitetura do Apache Streaming e grafo de linhagem de um D-Stream.

backup é lento no processo de recuperação. Ambas as abordagens não conseguem lidar com dados retardatários. A ideia do D-Stream é computar os dados em fluxo sem estado (*stateless*) utilizando computação em lote determinística em intervalos curtos de tempo. O D-Stream estrutura as computações da seguinte maneira: i) o estado em cada etapa temporal é totalmente determinístico ao dado de entrada, abolindo a necessidade de protocolos de sincronização e ii) a dependência entre o estado atual e o antigo são visíveis em uma fina granularidade. Isso garante ao Spark Streaming um excelente mecanismo de recuperação, similar aos de sistemas em lote e que supera a replicação e *upstream backup*. A latência do D-Stream é baixa, pois ele usa a estrutura de dados RDD, que computa os dados na memória e utiliza a abordagem de linhagem, conforme ilustrado na Figura 4.5(b). O D-Stream possui uma rápida recuperação de falhas e dados retardatários, pois utiliza o determinismo para prover um novo mecanismo de recuperação que é a recuperação paralela. Quando um nó falha, cada nó do *cluster* trabalha para computar novamente e recuperar os RDDs do nó em falha, resultando em uma recuperação significantemente mais rápida do que as outras duas abordagens [Zaharia et al., 2013]. O D-Stream pode recuperar retardatários utilizando execução especulativa [Dean e Ghemawat, 2008]. Cada D-Stream, assim como os RDDs, são imutáveis, assim cada transformação gera um novo D-Stream. Um D-Stream é uma sequencia de conjunto de dados imutáveis e particionadas (RDD) que podem ser influenciados pela transformação determinística.

4.4.2. Apache Flink

O Apache Flink [Carbone et al., 2015a] foi proposto em 2009 com o nome Stratosphere [Warneke e Kao, 2009]. Foi incubado na fundação Apache em 2014 e em dezembro do mesmo ano foi promovido a projeto *Top Level*. O Flink suporta processamento de fluxo e lote, tornando-se uma plataforma de processamento híbrida. A arquitetura do Flink pode ser distinguida em quatro camadas, Desenvolvimento, núcleo, APIs e bibliotecas, conforme apresentados na Figura 4.6(a).

O núcleo do Flink é o motor distribuído de fluxo de dados, que executam os programas desenvolvidos. As tarefas de analítica do Flink são abstraídas em grafos acíclico dirigido (GAD) formado por quatro componentes: fontes; operadores; torneiras de saída; e registros que percorrem o grafo. O grafo apresentado na Figura 4.7, consiste em: i) ope-



(a) Arquitetura do Flink. Adaptado [Carbone et al., 2015a]

(b) Modelo de processamento do Flink. Adaptado [Carbone et al., 2015a]

Figura 4.6. Arquitetura e modo de trabalho do Flink

radores de estado (*stateful*) e ii) fluxos de dados que representam dados produzidos por um operador e estão disponíveis para consumo pelos operadores. Como os grafos de fluxo de dados são executados de forma paralela, os operadores são paralelizados em uma ou mais instâncias, chamadas subtarefas, e fluxos são divididos em uma ou mais partições. Os operadores de estado, que podem ser sem estado, em casos especiais, implementam toda a lógica de processamento, como operações de `filter`, `map`, `hash join`, `window` etc. Os fluxos distribuem dados entre operadores de produção e consumo em vários padrões, como ponto a ponto, propagação (*broadcast*), repartição, difusão (*fan-out*) e mesclagem.

O Flink possui duas APIs bases, a API DataSet para processamento de dados estáticos e finitos, frequentemente associados a processamento em lote, e a API DataStream para processamento de dados dinâmicos e potencialmente ilimitados, frequentemente associada a processamento em fluxo. As bibliotecas do Flink são específicas para cada API e são FlinkML para aprendizado de máquina, Gelly para processamento de grafos e Table para processamento de operações SQL.

Um cluster do Flink trabalha no modelo mestre-escravo conforme mostrado na Figura 4.6(b) e é composto por três tipos de processos: o cliente, o Gerente de Trabalhos e pelo menos um Gerente de Tarefas. O cliente pega o código do programa Flink, o transforma em um gráfico de fluxo de dados e o envia para o gerente de trabalho. O gerente de trabalho coordena a execução distribuída do fluxo de dados. Ele rastreia o estado e o progresso de cada operador e fluxo, programa novos operadores e coordena os pontos de verificação (*checkpoints*) e a recuperação. Em uma configuração de alta disponibilidade, o gerente de trabalho persiste em um conjunto mínimo de metadados em cada ponto de verificação para um armazenamento tolerante a falhas, de modo que um gerente de trabalho em espera possa reconstruir o ponto de verificação e recuperar a execução do fluxo de dados a partir dele. O processamento de dados ocorre nos gerentes de tarefas. Um Gerente de tarefas executa um ou mais operadores que produzem fluxos e relatórios sobre seu status para o gerente de trabalho. Os gerentes de tarefas mantêm os *buffer* para armazenar ou materializar os fluxos e as conexões de rede para trocar os fluxos de dados entre os operadores [Carbone et al., 2015a]. A abstração do fluxo de

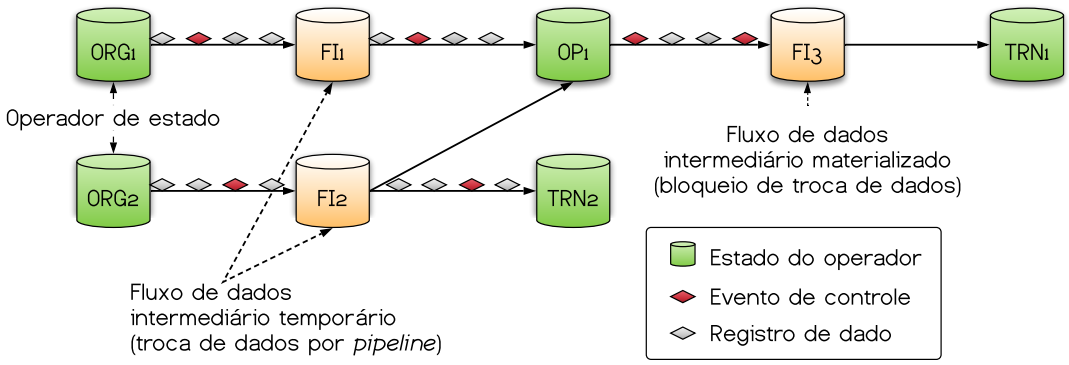


Figura 4.7. Grafo de fluxo de dados. Adaptado [Carbone et al., 2015a]

dados no Flink é chamada DataStream e é definida como uma sequência de registros parcialmente ordenados. Parcialmente, pois não há garantia de ordem caso um elemento operador receba mais de um fluxo de dados como entrada [Andreoni Lopez et al., 2018].

O Flink oferece execução confiável com garantias de consistência da semântica de processamento "exatamente uma" e lida com falhas com ponto de verificação e re-execução parcial. O mecanismo de verificação do Apache Flink baseia-se em captura momentânea do estado do sistema (*snapshot*) distribuídos para obter garantias de processamento exatamente uma vez. A natureza possivelmente ilimitada de um fluxo de dados torna a recomputação na recuperação impraticável, já que possivelmente meses de computação precisarão ser repetidos para uma tarefa longa. Para diminuir o tempo de recuperação, o Flink obtém uma captura de estado *snapshot* dos operadores, incluindo a posição atual dos fluxos de entrada em intervalos regulares. O mecanismo usado no Flink é chamado de Captura de estado de Barreiras Assíncronas (*Asynchronous Barrier Snapshotting* - ABS) [Carbone et al., 2015a]. As barreiras são registros de controle injetados nos fluxos de entrada que correspondem a um tempo lógico e separam logicamente o fluxo para a parte cujos efeitos serão incluídos na captura atual e a parte que será capturada posteriormente. Um operador recebe barreiras de *upstream* e primeiro executa uma fase de alinhamento, certificando-se de que as barreiras de todas as entradas foram recebidas. Então o operador escreve seu estado, como por exemplo conteúdos de uma janela deslizante ou uma estrutura de dados customizada em um sistema de arquivo durável, com o HDFS por exemplo. A recuperação de falhas reverte todos os estados do operador para seus respectivos estados retirados da última captura de estado bem-sucedida e reinicia os fluxos de entrada a partir da última barreira para a qual há uma captura de estado. A quantidade máxima de recálculo necessária na recuperação é limitada à quantidade de registros de entrada entre duas barreiras consecutivas. Além disso, a recuperação parcial de uma subtarefa com falha é possível, reproduzindo registros não processados armazenados em *buffer* nas subtarefas do fluxo ascendente.

4.5. Processamento de Grandes Massas de Dados em Fluxo

O modelo de processamento de fluxos prevê os dados chegando em linha (*online*) e, com isso, o sistema não possui o controle da ordem de chegada dos dados e, nem mesmo, da frequência com que os dados chegam. Os dados chegam de maneira contínua e ilimitada, sendo que o tamanho e tipo dos dados são desconhecidos. No processamento

de fluxos em tempo real a amostra é processada apenas uma vez, visto que o sistema tem de estar disponível a novos dados [Andreoni Lopez et al., 2018].

Amostras podem ser armazenadas temporariamente em memória, mas a memória é pequena em relação ao tamanho potencial dos dados que chegam nos fluxos. Dessa forma, para atingir o objetivo do processamento dos dados em tempo real, o processamento por fluxo impõe restrições ao tempo de processamento por amostra, já que se a taxa de chegada de dados for maior que a taxa de serviço de processamento, a fila de espera de dados a processar cresce indefinidamente e, como consequência, haveria descarte de dados. Portanto, técnicas para prover o processamento eficiente por fluxo constituem o uso de plataformas de processamento distribuído e o uso de técnicas de aproximação para agilizar o processamento dos dados.

Os métodos tradicionais de aprendizado de máquina se baseiam em sistemas que assumem que todos os dados coletados são totalmente carregados em um lote único, para então serem processados de maneira centralizada [Qiu et al., 2016]. Conforme ocorre o aumento do volume de dados, as técnicas existentes de aprendizado de máquina falham ao serem confrontadas com volumes inesperados de dados e, também, com o requisito de retornar a saída do processamento com a mesma agilidade que os dados são gerados. Assim, surge a necessidade de se desenvolver novos métodos de aprendizado de máquina mais rápidos e com comportamento adaptativo para atender às demandas do processamento de grandes massas de dados em tempo real [Costa et al., 2012].

Em muitos casos, certos padrões e rastros de comportamento estão escondidos no meio de um grande volume de informação e só são descobertos com auxílio de sistemas que utilizam aprendizado de máquina. Conforme novas informações são disponibilizadas, as estruturas de decisão devem ser revistas e atualizadas. Diversos modelos atualizam seus parâmetros considerando uma amostra por vez. Esses modelos são: aprendizado incremental, aprendizado *online*, aprendizado sequencial ou revisão da teoria [Gama et al., 2014]. Métodos incrementais não possuem restrição de tempo ou ordem das amostras, enquanto métodos *online* exigem que as amostras sejam processadas em ordem e apenas uma vez, de acordo com o tempo de chegada. Muitos algoritmos incrementais podem ser usados de maneira *online*, mas algoritmos que modelam comportamento no tempo necessitam que as amostras estejam em ordem.

Em ambientes dinâmicos e não estacionários, a distribuição de dados pode mudar ao longo do tempo, produzindo o fenômeno do desvio de conceito (*Concept Drift*). O desvio de conceito refere-se a mudanças na distribuição condicional da saída, ou seja, variável alvo, dado o vetor de características de entrada, enquanto a distribuição da entrada pode permanecer inalterada [Gama et al., 2014]. Um exemplo de mudança de conceito são os padrões de consumo de clientes. As preferências de compras podem mudar com o tempo, dependendo do dia da semana, disponibilidade de alternativas ou mudanças de salario. Em segurança, a criação de modelos para a detecção de ameaças pode ficar obsoleta com mínimas variações na composição dos ataques [Lobato et al., 2018]. A mudança de conceito afeta o desempenho da maioria dos algoritmos de aprendizado tornando-os menos precisos à medida que o tempo passa.

Um preditor eficaz deve ser capaz de rastrear essas mudanças e se adaptar rapidamente a elas. Um problema difícil ao lidar com a mudança de conceito é distinguir

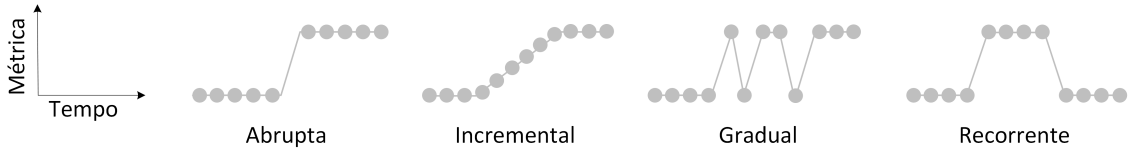


Figura 4.8. Tipos de mudanças de conceito. Em ambientes dinâmicos a distribuição de dados pode mudar ao longo do tempo, produzindo o fenômeno de mudança de conceito (*Concept Drift*).

entre uma mudança real e ruído. Alguns algoritmos podem reagir excessivamente ao ruído, interpretando-o erroneamente como mudança de conceito, enquanto outros podem ser altamente robustos ao ruído, ajustando-se às mudanças muito lentamente. Os quatro tipos diferentes de mudanças conhecidas são (Figura 4.8): i) mudança súbita ou abrupta; ii) mudança incremental; iii) mudança gradual; e iv) mudança recorrente ou cíclica.

Diferentes abordagens na detecção de mudança de conceito podem ser usadas dependo do domínio da classificação [Andreoni Lopez et al., 2019]. A primeira e a mais simples assume que os dados são estáticos e, portanto, não existe mudança na distribuição dos dados, treinando o modelo uma única vez e usando o mesmo modelo para dados futuros. Outra abordagem é atualizar periodicamente o modelo estático com dados históricos mais recentes, também conhecido como aprendizagem incremental. Alguns algoritmos de aprendizado de máquina como os algoritmos de regressão ou as redes neurais permitem avaliar a importância dos dados de entrada. Nesses algoritmos, é possível usar uma ponderação inversamente proporcional à história dos dados, de modo que seja dada maior importância aos dados mais recentes, com um peso maior, e menor aos dados menos recentes, com peso menor. Outra abordagem é o uso de algoritmos de conjunto de classificadores como AdaBoost ou Floresta Aleatória (*Random Forest*). Assim, o modelo estático é deixado intacto, mas um novo modelo aprende a corrigir as previsões do modelo estático com base nos relacionamentos de dados mais recentes. Finalmente, também é possível detectar mudanças de conceito utilizando heurísticas ou estatísticas intrínsecas do dados. Heurísticas como a acurácia ou a precisão são principalmente utilizadas em um cenário de aprendizado supervisionado no qual as etiquetas dos dados estão presentes durante o treinamento e a classificação. No entanto, a presença de etiquetas durante a classificação não é habitual em um ambiente de produção. No cenário de aprendizado não supervisionado, a comparação estatística de amostras de chegada, ou o agrupamento de amostras, com as amostras usadas para treinar o sistema, pressupõe que uma mudança de conceito é detectada sempre que novos grupos são encontrados [Jordaney et al., 2017]. Esses métodos de detecção de mudanças são propícios ao consumo de recursos, já que as medidas baseadas em distâncias são realizadas sobre as amostras obtidas.

4.5.1. Técnicas para a Mineração de Dados em Fluxo

Gaber *et al.* categorizam as soluções de tratamento de fluxos como baseadas em dados ou baseadas em tarefas [Gaber, 2012]. As soluções baseadas em dados têm como objetivo diminuir a representação dos dados, seja através de transformações horizontais, diminuindo o número de características a serem tratadas, ou transformações verticais, selecionando-se um subgrupo de observações a serem tratadas, também conhecido como amostragem. As soluções baseadas em tarefas concentram-se em uso de técnicas computacionais para tornarem as soluções eficientes tanto em relação a tempo quanto a espaço

de armazenamento. As técnicas baseadas em dados baseiam-se em resumir os dados ou escolher subconjuntos de dados do fluxo de entrada.

Amostragem de dados (*Sampling*). Enquanto algumas tuplas de dados são selecionadas para o processamento, outras são descartadas. No caso de chegada de dados a uma taxa superior ao que o sistema consegue processar em tempo hábil, a amostragem reduz a taxa de chegada de dados através do descarte de tuplas. Um possível cenário de uso para a amostragem de dados é na execução de funções de junção (*join*) em banco de dados. Uma operação de junção com amostragem retorna um resultado aproximado, mas sem realizar o bloqueio da base de dados e mais rapidamente que a operação completa. O algoritmo clássico para manter uma amostragem aleatória em linha é a técnica *recevoir sampling*. O algoritmo mantém uma amostra de tamanho *s*, chamada reservatório. Quando novos fluxos de dados chegam, cada novo elemento tem uma probabilidade de substituir um elemento antigo do reservatório. Uma extensão desse algoritmo é manter uma amostra de tamanho *k* sobre uma janela deslizante de tamanho *n* dos dados mais recentes.

Descarte de carga (*Load Shedding*) refere-se ao processo de descarte de sequências de fluxos de dados quando a taxa de entrada excede a capacidade de processamento do sistema. Assim, o sistema é capaz de ter um comportamento adaptativo para atingir os requerimentos de latência. Tal técnica também recai sobre a perda de informações. Essa técnica é geralmente utilizada em sistemas de consultas dinâmicas. Na mineração de dados, o descarte de carga é difícil de ser usado, pois o descarte de blocos de dados do fluxo pode gerar a perda de dados úteis para a construção de modelos ou pode descartar padrões de interesse a serem identificados em uma série temporal.

Rascunho (*Sketching*) é processo de projetar um subconjunto aleatório de atributos, ou domínio, dos dados. A ideia chave é produzir resultados mais rápidos com limites de erro comprovados matematicamente. O rascunho faz uma amostragem vertical, excluindo colunas de atributos, dos dados que chegam em fluxo. O processo de rascunho é aplicado em diferentes fluxos de dados e em consultas de agregação [Gaber, 2012]. A principal desvantagem é a perda de acurácia já que os resultados são aproximados. Uma alternativa ao uso do rascunho em aprendizado de máquina em fluxos de dados é a análise de componentes principais (*Principal Components Analysis*- PCA), em que ao invés de se usar um subconjunto dos atributos, usa-se uma combinação linear dos atributos que maximize a variância dos dados.

Estruturas de dados de Sinopse (*Synopsis*) são estruturas de resumo de dados em memória. A ideia central é gerar uma aproximação do resultado, com uma complexidade de memória menor. A proposta, por exemplo, *hash sketches* visa a criação de um vetor de bits com tamanho L, em que $L = log(N)$, sendo N o número de dados. Seja $lsb(y)$ a função que denota a posição do bit 1 menos significativo na representação binária de y. Os dados x que chegam são mapeados em uma posição do vetor de bits através de uma função *hash(x)*, uniforme, e da função $lsb(hash(x0)$, que marca no vetor de bits a ocorrência do dado. A partir dessa proposta, pode-se definir que sendo R a posição do valor zero mais à direita no vetor de bits, é possível estimar o número de elementos no vetor de bits como $E[R] = log(\phi d)$, em que $\phi = 0.7735$ e, então, $d = 2^R/\phi$. O uso de histogramas para estimar a frequência relativa de amostras em fluxo também constitui estruturas de dados de sinopse.

Agregação é a técnica de resumir os dados que chegam. O resumo pode ser feito na forma de média, variância, máximo ou mínimo. O custo de memória ao usar essa técnica é muito baixo, mas a técnica falha caso o fluxo de dados tenha muita variação. Vale ressaltar como técnicas de agregação o cálculo recursivo da média, dado por

$$\overline{x_i} = \frac{(i-1) \times \overline{x_{i-1}} + x_i}{i}, \tag{9}$$

em que x_i é média atual, após i rodadas, e o cálculo recursivo da variância, dado por

$$\sigma_i = \sqrt{\left(\sum x_i^2 - \left(\sum x_i\right)^2 / i\right) / (i-1)}. \tag{10}$$

Wavelets é uma transformada em que o sinal passa a ser representado por uma soma de formas de onda, mais simples e definidas na construção da transformada, em diferentes escalas e posições. A transformada *wavelet* tem como objetivo capturar tendências em funções numéricas, decompondo o sinal em um conjunto de coeficientes. De forma semelhante ao que ocorre com a redução de características usando PCA, os coeficientes mais baixos podem ser descartados. O conjunto reduzido de coeficientes é usado em algoritmos que operam sobre o fluxo. A transformada mais usada no contexto de algoritmos de fluxo é a *Haar wavelet*.

As técnicas para processamento de dados em fluxo baseadas em tarefas são métodos que modificam técnicas existentes, ou criam novas, para atender ao desafio computacional do processamento em fluxo.

Algoritmos de aproximação retornam resultados aproximados da computação limitados por limiares de erro [Gaber, 2012]. Na mineração de dados em fluxos, em especial, os algoritmos com resultados aproximados têm notoriedade, pois os resultados são esperados de serem gerados de maneira contínua, rápida e com recursos computacionais limitados.

Janelas de tempo são comumente usadas para resolver consultas em fluxos de dados sem fim definido. Ao invés de executar uma computação sobre o dado completo, a computação é realizada sobre um subconjunto de dados, eventualmente mais de uma vez sobre os mesmos dados. Nesse modelo, uma marcação de tempo (*timestamp*) é associado a cada dado entrante. A marcação de tempo é usada para definir se um dado está dentro ou fora da janela considerada. A computação é executada somente sobre os dados que estão dentro da janela considerada. Algumas abordagens distintas de janelas são a janela de ponto de referência (*landmark window*), janela saltitante, a janela deslizante (*sliding window*) e a janela enviesada (*tilted window*).

A janela de ponto de referência identifica pontos relevantes nos fluxos de dados e, a partir de então, calcula os operadores de agregação a partir desse ponto de referência. Janelas sucessivas compartilham o mesmo início e são de tamanhos crescentes. Um ponto de referência, por exemplo, pode ser o início de um novo dia para uma agregação de dados diária. A janela saltitante é uma estrutura que considera um determinado número fixo de amostras e quando um conjunto subsequente com o número suficiente de amostras chega, as anteriores são descartadas e computação passa a ser feita sobre o novo conjunto de amostras. A janela saltitante usa o conjunto de tamanho fixo de amostras apenas uma vez

e cada amostra só é usada em um conjunto. A janela deslizante, por sua vez, é uma estrutura de dados de tamanho fixo que se insere amostras mais recentes e retira as amostras mais antigas de modo similar ao modelo de *primeiro a chegar, primeiro a sair*. Tal estrutura é interessante computacionalmente, pois na maioria dos casos o passado total não é tão relevante quanto o passado recente. Assim, a janela deslizante considera somente um número fixo de amostras no passado mais recente. A estrutura da janela deslizante é bastante utilizada para o cálculo de médias móveis. Por fim, a janela enviesada cria diferentes resoluções para a agregação dos dados. Diferente das janelas anteriores em que as amostras ou estavam dentro da janela ou fora, na janela enviesada, as amostras mais recentes são tratadas com granularidade fina, enquanto amostras do passado são agrupadas em granularidades distintas. Quanto mais no passado, mais grossa a granularidade de agrupamento das amostras.

4.5.2. Métodos de Aprendizado de Máquina *Online*

Uma primeira abordagem para o tratamento de grandes massas de dados em fluxo é o uso de métodos de aprendizado que sejam capazes de aprender com dados infinitos em tempo finito. A ideia central é aplicar métodos de aprendizado que limitem a perda de informação ao usar modelos com dados finitos em relação a modelos com dados infinitos. Para tanto, a perda de informação é medida em função do número de amostras usadas em cada etapa de aprendizado e, então, o número de amostras usadas em cada etapa é minimizado mantendo-se o limiar de perda conservado.

A resolução do problema de o quanto de informação se perde ao diminuir o número de amostras é dada usando o limite de Hoeffding (*Hoeffding bound*) [Gama et al., 2014]. Considerando uma variável aleatória real x, cujo valor está contido no intervalo R, supõe-se que são feitas n observações independentes da variável e computa-se a média $\bar{r}$. O *Hoeffding bound* garante que, com probabilidade $1-\delta$, a média verdadeira da variável é dada por, pelo menos, $\bar{x}-\varepsilon$, em que

$$\varepsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2n}}. \tag{11}$$

O *Hoeffding bound* é independente da distribuição que gera a variável x. A partir desse resultado, desenvolvem-se os algoritmos de aprendizado de máquina para treinamento com fluxos de dados. Contudo, vale ressaltar que se assume que os valores gerados pela variável x advêm de um processo estocástico estacionário. Nos casos em que há uma mudança no processo que gera a variável usada no treinamento de métodos de aprendizado por fluxo é dito que ocorre uma mudança de conceito (*concept drift*) e, portanto, faz-se necessário um novo treinamento do método de aprendizado [Wang et al., 2013].

As abordagens típicas para aprender novas informações envolvem a manutenção do comportamento estocástico dos dados ou o descarte do classificador existente e, consequentemente, o retreinamento com os dados acumulados até o momento. As abordagens que consideram o fim da estabilidade estatística dos dados, o processo deixa de ser estocástico, resultam na perda de todas as informações adquiridas anteriormente, o que é conhecido como o esquecimento catastrófico. Dessa forma, Polikar *et al.* definem que algoritmos de aprendizado incremental devem satisfazer os requisitos de

obter informações adicionais de novos dados; de não exigir acesso aos dados originais, usados para treinar o classificador já existente; de preservar o conhecimento previamente adquirido, ou seja, não deve sofrer esquecimento catastrófico; e de ser capaz de acomodar novas classes que possam ser introduzidas por novos dados. Assim, classificadores que adotem o aprendizado incremental não requerem o treinamento de todo o classificador no caso de uma mudança no comportamento estacionário dos dados em fluxo.

Algoritmos em linha de árvores de decisão incrementais

Esses algoritmos são divididos em duas categorias: i) árvores construídas através de um algoritmo guloso de busca, em que a adição de novas informações envolvem a reestruturação completa da árvore de decisão e ii) árvores incrementais que mantêm um conjunto suficiente de estatísticas em cada nó da árvore para realizar um teste de divisão do nó, tornando a classificação mais específica, quando as estatísticas acumuladas no nó são favoráveis à divisão. Um exemplo desse tipo de árvore incremental é o sistema *Very Fast Decision Tree* (VFDT). O objetivo do sistema VFDT é projetar um método de aprendizado por árvore de decisão para conjuntos de dados extremamente grandes, potencialmente infinitos. A ideia central é que cada amostra de informação seja lida uma única vez e em um pequeno tempo de processamento. Isso possibilita o gerenciamento direto das fontes de dados em linha (*online*), sem armazenar as amostras. Para encontrar o melhor atributo que deve ser testado em um determinado nó, pode ser suficiente considerar apenas um pequeno subconjunto das amostras de treinamento que passam por esse nó. Assim, dado um fluxo de amostras, as primeiras serão usadas para escolher o teste da raiz; uma vez que o atributo da raiz é escolhido, as amostras subsequentes são transmitidas ao nós folhas correspondentes e usadas para escolher os atributos apropriados nesses nós, e assim por diante, recursivamente.

Em um sistema VFDT, a árvore decisão é aprendida recursivamente, substituindo folhas por nós de decisão. Cada folha armazena estatísticas suficientes sobre valores de atributos. Estatísticas suficientes são aquelas necessárias por uma função de avaliação heurística que avaliam o mérito de testes de divisão de nós baseados em valores de atributos. Quando uma amostra está disponível, ela atravessa a árvore da raiz até uma folha, avaliando o atributo apropriado em cada nó e seguindo o ramo correspondente ao valor do atributo na amostra. Quando a amostra chega à folha, as estatísticas são atualizadas. Então, as condições possíveis baseadas nos valores dos atributos são avaliadas. Caso haja suporte estatístico suficiente em favor de um teste de valor de um atributo em relação aos demais, a folha é convertida em nó de decisão. O novo nó de decisão vai ter tantos descendentes quantos valores possíveis para o atributo de decisão escolhido. Os nós de decisão mantêm somente as informações sobre o teste de divisão instalado no nó. O estado inicial da árvore consiste em uma folha única que é a raiz da árvore. A função de avaliação heurística é o Ganho de Informação (*Information Gain*), denotado por $H(\cdot)$. As estatísticas suficientes para estimar o mérito de um atributo nominal são os contadores n_{ijk} que representam o número de exemplos da classe k que chegam à folha, em que o atributo j recebe o valor i. O ganho de informação mede a quantidade de informação necessária para classificar uma amostra que chega ao nó: $H(A_j) = info(amostras) - info(A_j)$. A informação do atributo j é dada por

$$info(A_j) = \sum_i P_i \left(\sum_k -P_{ik} log_2(P_{ik}) \right), \tag{12}$$

em que $P_{ik} = \frac{n_{ijk}}{\sum_a n_{ajk}}$ é a probabilidade de se observar o valor de o atributo i dada a classe k e $P_i = \frac{\sum_a n_{ija}}{\sum_a \sum_b n_{ajb}}$ é a probabilidade de observar o valor do atributo i.

No sistema VFDT usa-se o limiar de Hoeffding, Equação 11, para decidir quantas amostras são necessárias observar antes de instalar um teste de separação em cada folha. Sendo $H(\cdot)$ a função de avaliação do atributo, para o ganho de informação, $H(\cdot)$ é o $log_2(||K||)$, em que K é o conjunto de classes. Seja x_a o atributo com maior valor de $H(\cdot)$, x_b o atributo com o segundo maior valor de $H(\cdot)$ e $\overline{\Delta H} = \overline{H}(x_a) - \overline{H}(x_b)$, a diferença entre os dois melhores atributos. Então, se $\overline{\Delta H} > \varepsilon$, com n amostras observadas na folha, o limiar de Hoeffding define com probabilidade $1-\delta$ que x_a é realmente o atributo com o maior valor na função de avaliação. Assim, a folha deve ser transformada em um nó de decisão que divide em x_a.

A avaliação da função para cada amostra pode ser muito custosa e, portanto, não é eficiente computar $H(\cdot)$ na chegada de cada nova amostra. A proposta VFDT só computa a função de avaliação de atributo quanto um número mínimo de amostras, definido pelo usuário, são observadas desde a última avaliação. Quando dois ou mais atributos têm os mesmos valores de $H(\cdot)$ continuamente, mesmo com um grande número de amostras, o limiar de Hoeffding não é capaz de decidir entre eles. Então, o VFDT introduz uma constante τ em que se $\overline{\Delta H} < \varepsilon < \tau$, então a folha é convertida em um nó de decisão e o teste de decisão é baseado no melhor atributo. Gama *et al.* generalizam o funcionamento do sistema VFDT para atributos numéricos [Gama et al., 2014].

Naive Bayes Incremental

Dado um conjunto de treinamento $\chi = (\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_n)$, em que $\mathbf{x} \in \mathbb{R}^D$ são amostras com $D-dimensionais$ no espaço de atributos e $y \in 1, ..., K$ são as classes correspondentes para um problema de classificação em K classes, formula-se o Teorema de Bayes como

$$p(y = i|x) = \frac{p(i)p(\mathbf{x}|i)}{p(\mathbf{x})}, \tag{13}$$

em que $p(i)$ é a probabilidade *a priori* de ocorrência de uma amostra da classe e $p(y|\mathbf{x})$ é a distribuição de probabilidades desconhecida do espaço de atributos $\mathbf{x}$ e marcada com a classe i. Uma estimativa para a distribuição desconhecida é assumir a independência dos atributos dada a marcação da classe, levando a

$$p(x_1, x_2, ..., x_D|i)\ p(x_1|i)p(x_2|i)...p(x_D|i), \tag{14}$$

em que x_d representa a d-ésima dimensão no vetor de atributos $\mathbf{x}$. Assim, o classificador bayesiano é descrito como

$$F(\mathbf{x}) = arg\ \max_{i} \prod_{d=1}^{D} p(x_d|i). \tag{15}$$

Assim, a classificação é calculada fazendo a multiplicação de todas as probabilidades das classes para o valor dos atributos da amostra [Godec et al., 2010].

A versão incremental do classificador bayesiano prevê a atualização dos valores das probabilidades das classes por atributos conforme novas amostras são processadas. Uma abordagem para permitir o armazenamento eficiente das funções de probabilidade conforme as amostras chegam é realizar a discretização e armazenar histogramas dos atributos. A proposta IFFD (*incremental flexible frequency discretization*) apresenta um método para discretização de atributos quantitativos em uma sequência de intervalos de tamanhos flexíveis. Essa abordagem permite a inserção e a divisão de intervalos.

Aprendizado Incremental por Agregados de Classificadores

O algoritmo ARTMAP baseia-se na geração de novos agrupamentos de decisão em resposta a novos padrões que são suficientemente diferentes de instâncias vistas anteriormente. O valor de quanto diferente é um padrão já conhecido de um novo é controlado por um parâmetro de vigilância definido pelo usuário. Cada agrupamento aprende em um hiper-retângulo que é uma porção diferente do espaço de características, em um modo não supervisionado, que são então mapeados para classes alvo. Agrupamentos são sempre mantidos, o ARTMAP não sofre o esquecimento catastrófico. Além disso, ARTMAP não requer acesso a dados previamente vistos e pode acomodar novas classes. Contudo, o ARTMAP é muito sensível à seleção do parâmetro de vigilância, aos níveis de ruído nos dados de treinamento e à ordem em que os dados de treinamento chegam.

O algoritmo AdaBoost (*adaptive boosting*) gera um conjunto de hipóteses e as combina através da votação da maioria ponderada das classes previstas pelas hipóteses individuais. As hipóteses são geradas pelo treinamento de um classificador fraco[3], usando instâncias extraídas de uma distribuição atualizada periodicamente dos dados de treinamento. Esta atualização de distribuição garante que instâncias mal classificadas pelo classificador anterior sejam mais provavelmente incluídas nos dados de treinamento do próximo classificador. Assim, os dados de treinamento de classificadores consecutivos são voltados para instâncias cada vez mais difíceis de classificar.

O algoritmo de aprendizagem incremental Learn ++ é inspirado pelo AdaBoost, originalmente desenvolvido para melhorar o desempenho de classificação de classificadores fracos. Em essência, Learn ++ gera um conjunto de classificadores fracos, cada um treinado usando uma distribuição diferente de amostras de treinamento. As saídas desses classificadores são então combinadas usando o regime de votação por maioria para obter a regra final de classificação. O uso de classificadores fracos é interessante, pois a instabilidade para que construam suas decisões é suficiente para que cada decisão seja diferente das demais, para que pequenas modificações em seus conjuntos de dados de treinamento sejam refletidas em classificações distintas.

[3] Algoritmos de classificação que a acurácia é próxima à classificação aleatória.

K-Médias Incremental

O algoritmo de agrupamento k-médias executa um critério de otimização iterativo da soma das distâncias ao quadrado de todos os pontos ao centro de cada agrupamento. No caso de um problema de agrupamentos por aprendizado de máquina não-supervisionado, define-se que N é o conjunto de entradas $\overline{x}_1, \overline{x}_2, ..., \overline{x}_N$, em que $\overline{x}_i \in R^n$. O problema consiste, então, em encontrar $K << N$ agrupamentos $c_1, c_2, ..., c_k$ com centros em $\overline{\omega}_1, \overline{\omega}_2, ..., \overline{\omega}_k$, respectivamente, tal que a distância D da soma ao quadrado das distâncias dos dados aos centros dos agrupamentos a que pertencem seja mínima. A distância D é equivalente ao erro médio quadrático (*Mean Sqaure Error* – MSE) e é dada por

$$D = \frac{1}{N} \sum_{j=1}^{K} \sum_{\overline{x} \in c_j} (\overline{x} - \overline{\omega}_j)^2. \tag{16}$$

As iterações do algoritmo clássico consistem em uma inicialização, em que são escolhidos K centros, e os elementos são classificados pela regra do vizinho mais próximo. Após, os centros dos agrupamentos são atualizados através do cálculo do centroide de cada agrupamento. Nas iterações seguintes, os dados são reclassificados de acordo com os novos centroides calculados. As iterações são executadas até a convergência do algoritmo quando os centros dos agrupamentos calculados na iteração i são idênticos aos de $i+1$.

A variação do algoritmo das k-médias para o tratamento sequencial dos dados consiste em recalcular os centros dos agrupamentos sempre que uma nova amostra chega. Essa variação depende de todos os dados já processados serem acessados novamente para o recálculo dos centros dos agrupamentos, o que gera uma demanda de recursos computacionais substancialmente alta, inviabilizando o seu uso. A variação do algoritmo usando blocos sequenciais, por sua vez, viabiliza a utilização do algoritmo de k-médias *online*, pois o agrupamento é realizado sobre blocos de dados acumulados. Cada bloco é usado por l épocas do algoritmo de k-médias e os resultados dos centros dos agrupamentos do bloco i são usados como os centros iniciais da iteração sobre o bloco $i+1$. A variação incremental do algoritmo define que o bloco é usado somente uma vez [Aaron et al., 2014]. A validade do resultado do algoritmo incremental reside no fato de que a distribuição de probabilidades dos atributos das amostras não muda, ou muda lentamente, entre blocos.

4.5.3. Aprendizado por Reforço

O aprendizado por reforço (*Reinforcement Learning* - RL) baseia-se em um agente interagindo com o ambiente, aprendendo uma política ótima, por tentativa e erro, para problemas de tomada de decisão sequenciais em campos de ciências naturais e sociais e engenharia [Li, 2018]. O agente de aprendizado por reforço interage com o ambiente através do tempo. A cada passo t de tempo, o agente aplica uma política $\pi(a_t|s_t)$, em que s_t é o estado que o agente recebe de um espaço de estados S e a_t é uma ação selecionada pelo agente em um espaço de ações A. Dessa forma, o agente mapeia o estado s_t em uma ação a_t para receber uma recompensa, ou penalidade, escalar r_t e realizar a transição para o próximo estado s_{t+1}. A transição ocorre de acordo com a dinâmica ou o modelo do ambiente. A função $R(s,a)$ modela a recompensa do agente e a função $P(s_{t+1}|s_t,a_t)$ modela a probabilidade de transição de estados do agente. O agente continua a execução do ciclo até atingir um estado terminal, quando o ciclo é reinicado. O retorno $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ é

o acumulo de recompensa descontada por um fator $\gamma \in [0,1)$. A ideia central do aprendizado por reforço é que o agente maximize a expectativa de um retorno de longo prazo para cada estado. Vale ressaltar que os mecanismos de aprendizado por reforço assumem que o problema satisfaz a propriedade de Markov, em que o futuro depende apenas do estado atual e da ação e não do passado. Assim, o problema é formulado como um Processo de Decisão de Markov (*Markov Decision Process* - MDP), definido pela 5-tupla (S,A,P,R,γ).

A aprendizagem de diferença temporal é um dos pilares do aprendizado por reforço, já que a aprendizagem de diferença temporal se refere aos métodos de aprendizagem para avaliação da função de valor, SARSA e *Q-learning*. O aprendizado de diferença temporal aprende a função de valor $V(s)$ diretamente da experiência com o erro da diferença temporal, com inicialização independente do modelo, *online* e totalmente incremental. O aprendizado de diferença temporal é um problema de previsão. A regra de atualização é $V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$, em que α é a taxa de aprendizagem e $r + \gamma V(s') - V(s)$ é o chamado erro de diferença temporal. O uso do algoritmo do gradiente descendente generalizado garante a convergência dessa gama de problemas [Lobato et al., 2018].

O problema de previsão ou avaliação de política com aprendizado por reforço consiste em calcular o estado ou função de valor de ação para uma política. O problema de controle é encontrar a política ótima. O algoritmo SARSA avalia a política com base em amostras da mesma política e refina a política através de uma metodologia gulosa em relação aos valores de ação. Em métodos fora da política (*off-policy*), o agente aprende uma função de valor ou política ideal, talvez seguindo uma política comportamental não relacionada. O algoritmo *Q-learning*, por exemplo, tenta encontrar diretamente valores de ação para a política ótima, não necessariamente se ajustando à política que gera os dados. Assim, a política encontrada pelo *Q-learning* é geralmente diferente da política que gera as amostras [Li, 2018]. As noções de métodos na política e fora da política relacionam-se com as ideias de avaliar as soluções encontradas com as mesmas políticas que geram os dados ou avaliar com políticas ligeiramente diferentes. Avaliar com políticas diferentes refere-se a usar um modelo para gerar os dados, enquanto em métodos sem modelo o agente aprende através de tentativas e erro nas ações tomadas. Os algoritmos de aprendizado por reforço podem executar em modo *online* ou *offline*. No modo online, os algoritmos de treinamento são executados em dados adquiridos em fluxos sequenciais. No modo offline ou em lote, os modelos são treinados em todo o conjunto de dados.

O aprendizado por reforço é uma estratégia presente em trabalhos que realizam a extração de conhecimento e a tomada de decisões em redes sem fio [Liu e Yoo, 2017, Tabrizi et al., 2011, Chen e Qiu, 2011]. Liu e Yoo propõe um esquema que usa o algoritmo Q-learning para alocar dinamicamente sub-quadros em branco para que ambos LTE-U e Sistema WiFi pode coexistir com sucesso [Liu e Yoo, 2017]. A técnica de ajuste é baseada no aprendizado por reforço. A proposta introduz uma nova estrutura de quadro LTE-U, que além de alocar subquadros em branco, também reduz o atraso de LTE. Na proposta as ações do algoritmo são modeladas através de uma tupla que contém o número total subquadros em um bloco de quadros e a porção de subquadros para LTE-U. Os estados são modelados como o conjunto de todas as ações tomadas até aquele estado. Tabrizi *et al.* advogam que a coexistência de várias tecnologias sem fio, as comunicações de próxima

geração tendem a usar um sistema integrado de redes, em que pontos de acesso WiFi e estações base da rede de celular trabalham em conjunto para maximizar a Qualidade de Serviço (QoS) do usuário móvel [Tabrizi et al., 2011]. Dessa forma, os autores propõem que os dispositivos móveis com diferentes tecnologias de acesso selecionem redes e alternar facilmente de um ponto de acesso ou estação base para outro para melhorar o desempenho do usuário. A proposta baseia-se na seleção de rede com o objetivo de maximizar a QoS, formulada como um Processo de Decisão de Markov e, portanto, um algoritmo baseado no aprendizado por reforço obtido para selecionar a melhor rede baseada na carga de rede atual e também nos possíveis futuros estados de rede. Chen e Qiu propõem uma abordagem para o sensoriamento cooperativo de espectro para rádios definidos por software baseado no algoritmo *Q-learning* [Chen e Qiu, 2011]. No sensoriamento do espectro cooperativo usando Q-learning, o conjunto de estados é composto por todas as combinações de um vetor de bits em que cada usuário marca um bit e o conjunto de ações é $0, 1$, em que "0" significa que o canal está no estado "disponível" para usuários secundários e uma ação com índice "1" significa que o canal está em "ocupado" e indisponível usuários secundários. Atribuir valores para recompensa r para Q-learning pode ser complicado. No algoritmo proposto, as recompensas são positivas quando a ação está de acordo com a ocupação do canal e negativas em caso contrário.

4.5.4. Aprendizado Profundo - *Deep Learning*

O aprendizado profundo (*deep learning*) consistem em uma classe de técnicas de aprendizado de máquina que exploram várias camadas de processamento de informações não lineares para extração e transformação de recursos, usando aprendizado supervisionados ou não supervisionados, e para análise e classificação de padrões [Deng e Yu, 2014]. Assim, o aprendizado profundo baseia-se em aprender vários níveis de representação e abstração que ajudam a compor o sentido dos dados. Em especial, essa classe de técnicas está na interseção de áreas de pesquisas de redes neurais, inteligência artificial, modelagem gráfica, otimização, reconhecimento de padrões e processamento de sinais e, portanto, são geralmente usadas no processamento de imagens, som e texto [Kwon et al., 2017]. Os principais aspectos das técnicas que se caracterizam por serem de aprendizado profundo são os modelos que consistem de múltiplas camadas ou estágios de processamento não-lineares e os métodos para aprendizagem supervisionada ou não supervisionada de representação de características em camadas sucessivamente mais altas e mais abstratas. Dessa forma, a ideia chave do aprendizado profundo é gerar novas representações dos dados a cada camada, aumentando o grau de abstração da representação. A popularidade crescente das técnicas de aprendizado profundo deve-se ao aumento acelerado da capacidade de processamento de *chips*, como os das unidades gráficas (GPUs); ao aumento significativo dos dados disponibilizados para o treinamento dos modelos; e aos avanços nas pesquisas de aprendizado de máquina [Kwon et al., 2017], que permitiram a exploração de funções complexas não-lineares de composição, a aprendizagem distribuída e hierárquica e uso efetivo de dados rotulados e não rotulados.

As arquiteturas e técnicas de aprendizado profundo podem ter uso voltado para a síntese/geração ou reconhecimento/classificação e, assim, são geralmente classificadas [Deng, 2014]. **Arquiteturas profundas geradoras** caracterizam as propriedades de correlação de alta ordem dos dados observados para a análise ou síntese de padrões e/ou

caracterização das distribuições estatísticas conjuntas dos dados observados e de suas classes associadas. **Arquiteturas profundas discriminativa** proveem valores para realizar a discriminação em classes de padrões e, por vezes, caracterizando as distribuições *a posteriores* de classes condicionadas aos dados observados. **Arquiteturas híbridas profundas** têm por objetivo discriminar dados em classes, mas é assistida com os resultados de arquiteturas geradoras através da otimização e/ou regularização, ou quando critérios discriminativos são usados para aprender os parâmetros em modelos generativos. As arquiteturas geradoras estão associadas à identificação e ao reconhecimento de padrões ocultos nos dados observados, enquanto as arquiteturas discriminativas estão associadas à classificação de dados observados em classes definidas. Ressalta-se que as arquiteturas geradoras relacionam-se a problemas de aprendizado não-supervisionado, enquanto as arquiteturas discriminativas, aos problemas de aprendizado supervisionado. No processo de aprendizado não supervisionado não existem dados rotulados e, assim, principal objetivo desse processo é gerar dados rotulados aplicando algoritmos de aprendizado não supervisionados, como Máquina Restrita de Boltzmann (*Restricted Boltzmann Machine* - RBM), rede de crença profunda (*Deep Belief Network* - DBN), rede neural profunda (deep neural network - DNN), *AutoEncoders* generalizados, rede neural recorrente (*recurrent neural network* - RNN) [Kwon et al., 2017].

As Máquinas Retritas de Boltzmann são modelos generativos probabilísticos capazes de extrair automaticamente características dos dados de entrada usando um algoritmo de aprendizado completamente não supervisionado [Kim et al., 2010]. As RBMs consistem em uma camada oculta e uma camada de neurônios visíveis com conexões entre os neurônios ocultos e visíveis representados por uma matriz de pesos. Para treinar uma RBM, as amostras de um conjunto de treinamento são usadas como entrada para a RBM através dos neurônios visíveis e a rede gera amostras alternadamente para trás e para frente entre os neurônios visíveis e ocultos. O objetivo do treinamento é aprender pesos das conexão entre neurônios visíveis e ocultos e aprender os vieses de ativação de neurônios, de modo que o RBM aprenda a reconstruir os dados de entrada durante a fase em que os neurônios visíveis dos neurônios ocultos são amostrados. Cada processo de amostragem é essencialmente uma multiplicação de matrizes entre um lote de amostras de treinamento e a matriz de peso, seguido por uma função de ativação de neurônio, que em muitos casos é uma função sigmóide $(1/(1+e^{-x}))$. A amostragem entre as camadas oculta e visível é seguida por uma ligeira modificação nos parâmetros, definida pela taxa de aprendizagem α, e repetida para cada lote de dados no conjunto de treinamento, e por tantas épocas quantas forem necessárias para alcançar a convergência [Kim et al., 2010].

A rede de crença profunda (DBN) consiste em múltiplas camadas de variáveis estocásticas e ocultas e relaciona-se com a RBM, pois consiste na composição e o empilhamento de várias RBMs. A composição de várias RBM permite que muitas camadas ocultas treinem dados de forma eficiente por meio de ativações de um RBM para estágios adicionais de treinamento [Kwon et al., 2017, Kim et al., 2010].

As Redes Neurais de Convolução (CNNs) representam uma das formas mais comuns de Redes Neurais Profundas (DNN) [Sze et al., 2017]. As CNNs são compostas por múltiplas camadas convolucionais e cada camada gera um mapa de características, uma abstração de nível mais alto dos dados de entrada, que preserva informações essenciais e únicas. Cada uma das camadas convolucionais na CNN é composta principalmente por

convoluções de alta dimensão, em que as ativações de entrada de uma camada são estruturadas como um conjunto de mapas de características de entrada, chamado de canal. Por esse motivo, as CNNs são geralemente usadas em aplicações de processamento de sinais. Cada canal é convolvido com um filtro distinto da pilha de filtros. O resultado dessa computação são as ativações de saída que compõem um canal de mapa de características de saída. Por fim, vários mapas de características de entrada podem ser processados juntos em lote para melhorar potencialmente a reutilização dos pesos do filtro.

AutoEncoder tem sido tradicionalmente usado para redução de dimensionalidade e aprendizado de características. A ideia fundamental do *AutoEncoder* a presença de uma camada oculta h que se refere à entrada e outras duas partes principais, a função de codificação $h = f(x)$ e decodificação ou reconstrução $r = g(h)$. O principal objetivo deste conceito é que tanto o codificador quanto o decodificador sejam treinados juntos e a discrepância entre os dados originais e sua reconstrução possa ser minimizada. O *AutoEncoder* profundo é uma parte de um modelo não supervisionado [Kwon et al., 2017].

Redes neurais convencionais baseiam-se no princípio de que todos os pontos de dados são independentes. Por esse motivo, se os pontos de dados estiverem relacionados a tempo ou espaço, a chance de perder o estado da rede é alta. Já as redes neurais recursivas (RNN) são baseadas em sequências, de modo que elas podem modelar uma entrada ou saída que seja composta de elementos independentes em sequência. A RNN pode ser usada em aprendizado não supervisionado ou supervisionado. Quando usada em aprendizado não supervisionado, a predição da sequência de dados da amostras de dados anteriores é possível, mas apresenta dificuldade de treinamento [Kwon et al., 2017].

O aprendizado profundo é normalmente treinado com a abordagem de gradiente estocástico descendente [Lobato et al., 2018], em que um exemplo de treinamento com o rótulo conhecido por vez é usado para atualizar os parâmetros do modelo. A estratégia pode ser usada para o aprendizado em fluxo (*online*). Contudo, uma estratégia para acelerar a aprendizagem consiste em realizar as atualizações em minilotes de dados (*mini-batches*) em vez de proceder sequencialmente com uma amostra de cada vez [Chen e Lin, 2014]. As amostras em cada minilote são o mais independentes possível e essa abordagem fornece um bom equilíbrio entre o consumo de memória e o tempo de execução.

Trabalhos recentes usam o aprendizado profundo para inferir características e comportamentos de redes sem fio. A proposta BiLoc desenvolve um algoritmo baseado em aprendizado profundo para explorar dados bi-modais, ângulo de fase estimado de chegada e amplitudes médias na interface de rádio de 5 GHz para redes sem fio, para gerar impressões digitais de localizações internas (*indoor*) de dispositivos [Wang et al., 2017]. O aprendizado profundo produz impressões digitais baseadas em características a partir dos dados bi-modais no estágio de treinamento off-line. Os pesos na rede de *autoencoder* profundo são as impressões digitais baseadas em características para cada posição. Wang *et al.* comparam duas abordagens de localização interna usando aprendizado profundo, uma com *autoencoders* e outra com redes neurais convulucionais [Wang et al., 2018]. Os autores concluem que a abordagem baseada em *autoencoders* apresentam menor erro na inferência da posição interna. Turgut *et al.*, por sua vez, usam *autoencoder* profundo para realizar a localização interna de dispositivos sem fim, porém consideram como características inicias a potência de sinal recebido de 26 pontos de acesso [Turgut et al.,

2019]. Wang *et al.* usam o aprendizado profundo para o reconhecimento de atividade mais preciso e robusto em canais de redes sem fio. A ideia central é selecionar ativamente canais WiFi disponíveis com boa qualidade e alternar entre canais adjacentes para formar um canal estendido. Wang *el al.* buscam por padrões sequenciais de uso de canal, então adotam um modelo de uma rede neural recursiva [Wang et al., 2018].

4.6. Discussão, Tendências e Desafios de Pesquisa

As principais aplicações baseadas em atuação automática para monitoramento de redes sem fio consistem em detecção de anomalia, geração de perfis de usuários, aplicações, estações e rede, abordagens baseadas em comportamento, visualização e otimização [Li et al., 2013]. Li *et al.* apontam que as principais aplicações de processamento de dados em fluxo de redes são o monitoramento de aplicações, estações e rede; a classificação do comportamento de aplicações, a segurança e a detecção de intrusão em redes. O monitoramento de rede fornece informações sobre os elementos de rede e a visão consolidada da rede é usada detectar problemas e propor soluções eficientes. O monitoramento de aplicações fornece informações sobre o uso na rede permite planejamento e alocação de recursos. O monitoramento de estações fornece informações sobre o padrão de utilização do usuário, da rede e da aplicação e é usado para planejamento da rede, do controle de acesso e para a verificação de violações de políticas de segurança. A classificação de do comportamento de aplicações permite identificar qual o tipo da aplicação em execução, mesmo quando o tráfego é criptografado. Outra a aplicação de dados em fluxo é a inferência da identidade de usuários baseada somente no comportamento dos fluxos de comunicação na rede. Por fim, Li *et al.* ainda ressaltam o uso do processamento de dados em fluxo como uma componente importante para prover segurança em rede através da identificação de anomalias ou da classificação de tráfego em busca de assinaturas de *worms*, varredura de portas, execução de *botnets*, ataques de negação de serviço, como também para a validação de novas políticas de rede.

Há ainda desafios de pesquisa para o tratamento de grandes massas de dados em tempo real. Os principais desafios do processamento em fluxo de dados para redes sem fio relacionam-se com as cinco dimensões principais no processamento de grandes massas de dados, volume, velocidade, variedade, veracidade e valor [Chen e Lin, 2014].

Estimação de erros nos dados relaciona-se com a granularidade das medições de realizadas por ferramentas de montiramento de redes, tais como SNMP, Netflow e OpenFlow. Erros podem acontecer nos processos de amostragem, transporte ou coleta e, portanto, associam-se à veracidade dos dados. Métodos para a estimação de intervalos de confiança, boas práticas na amostragem e a caracterização de erros mais comuns têm sido estudados e propostos para mitigar e efeito dos erros dos dados amostrados nas ações de controle e gerenciamento da rede [Li et al., 2013].

Significado dos dados relaciona-se ao valor dos dados e ao fato de que a maioria dos dados está dispersa em diferentes fontes e, portanto, o significado de dados coletados das várias fontes pode ser ligeiramente distinto. Isso pode afetar significativamente a qualidade dos resultados do aprendizado de máquina. Ontologia, web semântica e outras técnicas são propostas para mitigar problemas com o significado dos dados. Com base na modelagem de ontologias e na derivação semântica, padrões ou regras valiosos podem

ser descobertos, mas as técnicas de ontologias e web semântica ainda não estão maduras o suficiente.

Treinamento para o reconhecimento de padrões consiste em usar padrões rotulados para treinar os algoritmos de aprendizagem. Contudo, obter os rótulos implica custos de tempo e de computação, particularmente para os dados em fluxo em larga escala, sendo impactado pelo volume e pela velocidade em que os dados são criados. Em alguns casos, obter os padrões pode ser intratável computacionalmente. Outro desafio relacionado ao treinamento é o equilíbrio entre custo e precisão, chamado sobreajuste (*overfitting*), que é outro problema crítico em aberto.

Técnicas de integração de dados consistem na agregação de técnicas de mineração de dados, descoberta de conhecimento, computação em nuvem e aprendizado de máquina. Relaciona-se com a variedade dos dados, com a velocidade que novos dados são criados e com o volume de dados que chegam em fluxo. Contudo, cada técnica tem vantagens e desvantagens. Assim, uma tendência de pesquisa é a adoção de abordagens híbridas compostas de diversas técnicas.

Conjuntos de dados padrão e ambientes para pesquisa são é essenciais para identificar alguns problemas comuns em redes sem fio, juntamente com dados rotulados ou não rotulados para abordagens baseadas em aprendizado supervisionado ou não supervisionado. Para abordagens baseadas em aprendizado de reforço, devem ser construídos problemas de controle de rede padrão em conjunto com ambientes bem definidos a fim de permitir a comparação entre proposta de pesquisas [Sun et al., 2018].

Segurança e privacidade dos dados são desafios atuais para o processamento de grandes massas de dados, pois com o uso de tecnologias de mineração de dados e de aprendizado de máquina para analisar informações pessoais, é possível produzir resultados demasiadamente relevantes e interconectados que prejudicam a privacidade dos indivíduos. Informações pessoais que não são fornecidas *a priori* são inferidas através da correlação de dados provenientes de fontes distintas. Um dos principais desafios do processamento de grandes massas de dados é definir métodos eficientes e eficazes que gerem conhecimento preciso, preservando o desempenho da mineração e do aprendizado, ao passo que garantem a proteção das informações pessoais sensíveis;

Realização e aplicações consistem no uso do conhecimento gerado através do processamento em fluxo de grandes massas de dados em aplicações de tempo real. Em redes sem fio de grande porte, essas aplicações são temas de pesquisas recentes e relacionam-se com o controle da rede através da escolha de canal, mudança na associação entre clientes e pontos de acesso e na geração de perfis de uso da rede.

Outros desafios ainda abertos são identificados por Sun *et al*. Para a implementação de técnicas de aprendizado de máquina supervisionadas e não supervisionadas nas redes sem-fio, é essencial a criação de conjuntos de dados rotulados/não rotulados. Para abordagens baseadas em aprendizado de reforço, devem ser construídos problemas de controle de rede em ambientes bem definidos [Sun et al., 2018]. A transferência de aprendizado promete transferir o conhecimento aprendido de uma tarefa para outra tarefa semelhante. Evitando treinar modelos de aprendizado a partir do zero, o processo de aprendizado em novos ambientes pode ser acelerado, e o algoritmo aprendizado de

máquina pode ter um bom desempenho, mesmo com uma pequena quantidade de dados de treinamento. Portanto, a transferência de aprendizado é fundamental para a implementação prática de modelos de aprendizagem considerando o custo para treinamento sem conhecimento prévio. Usando o aprendizado de transferência, os operadores de rede podem resolver problemas novos, mas semelhantes, de maneira econômica.

Por outro lado, no gerenciamento da rede, o controle de *backhaul/fronthaul* heterogêneos baseado em aprendizado de máquina podem ser aplicados. Em futuras redes sem fio, várias soluções de *backhaul/fronthaul* coexistirão, incluindo fibra e cabo assim como sem-fio como a banda sub-6 GHz [Sun et al., 2018]. Cada solução consome quantidades de energia e largura de banda diferente. Portanto, para um bom desempenho do sistema, as técnicas baseadas em aprendizado de máquina podem ser usadas para selecionar soluções de *backhaul/fronthaul* adequadas com base nos padrões de tráfego extraídos e nos requisitos de desempenho dos usuários. Ainda no gerenciamento da rede, futuras atualizações das infraestruturas de rede sem fio devem ser desenvolvidas baseadas em técnicas de otimização amparadas por aprendizado de máquina.

Em outra vertente, o fatiamento da rede (*network slicing*) é defendido como uma maneira econômica de suportar diversos casos de uso. A ideia principal do fatiamento da rede é alocar recursos apropriados, incluindo recursos de computação, armazenamento em cache, *backhaul/fronthaul* e rádio sob demanda, para garantir os requisitos de desempenho de diferentes fatias isoladas. O fatiamento da rede pode se beneficiar do aprendizado de máquina quanto o mapeamento das demandas de serviço sobre os planos de alocação de recursos e, também, ao empregar transferência de aprendizado, o conhecimento sobre planos de alocação de recursos para diferentes casos de uso em um ambiente pode agir como conhecimento útil em outro ambiente, acelerando o processo de aprendizagem.

Projetos de pesquisa focam tanto no processamento em fluxo de grandes massas de dados, como no desenvolvimento de redes de experimentação de larga escala baseadas na tecnologia de redes sem fio. Alguns dos projetos de pesquisa focados em criar redes de experimentação sem fio abordados incluem o ORBIT, o NITOS, o OneLab, o FIT, o FUTEBOL e o FIBRE. Essas infraestruturas de testes permitem a experimentação em redes federadas incluindo diferentes tipos de dispositivos sem fio. Suas vantagens e desvantagens para aplicações de processamento em fluxo são avaliadas e discutidas.

OneLab[4] é uma iniciativa europeia que propõe a federação de ambientes de experimentação com diferentes tecnologias de acesso. O OneLab agrega ambientes de experimentação voltado para Internet das Coisas (IoTLab), um ambiente de experimentação de redes sem fio com nós Wi-Fi a/b/g/n (NITOS) e o ambiente PlanetLab europeu (PLE). **FIT**[5] é uma infraestrutura de testes aberta em larga escala para sistemas e aplicativos em comunicações sem fio e sensores. A FIT oferece uma grande variedade de tecnologias, tais como Internet das Coisas, redes sem fio e computação em nuvem, e também uma interface única de acesso ao sistema e um grande número de ferramentas de configuração e monitoramento. **FUTEBOL**[6] é um projeto de cooperação entre o Brasil e a Europa para desenvolver e implantar uma infraestrutura de experimentação que possibilite pesquisas

[4]Disponível em https://onelab.eu/.
[5]Disponível em https://fit-equipex.fr/.
[6]Disponível em http://www.ict-futebol.org.br/.

sobre o ponto de convergência entre redes ópticas e sem fio. **FIBRE**[7] é uma infraestrutura de experimentação que funciona como um laboratório virtual de larga escala para novas aplicações e modelos de arquitetura de rede. Organiza-se como uma federação de 11 ilhas de experimentação locais, entre as quais há ilhas de experimentação de redes sem fio, como, por exemplo, a disponível na Universidade Federal Fluminense (UFF). **PrEstoCloud**[8] foca em desenvolver pesquisas nas tecnologias de computação em nuvem e análise de dados em tempo real para fornecer uma arquitetura dinâmica e distribuída para gerenciamento proativo de recursos em nuvem, visando alocar o processamento na borda extrema da rede para reduzir a latência no processamento. PrEstoCloud combina a pesquisa de Big Data, computação em nuvem e computação em nuvem em tempo real. **Metron**[9] é um arcabouço de análise de segurança baseado no processamento de grandes massas de dados. A ideia chave desse arcabouço é permitir a correlação de eventos de segurança originados de fontes distintas e, para tanto, o arcabouço emprega como fonte de dados sensores na rede, registros de ações (*logs*) de elementos ativos de segurança em rede e fontes de telemetria. **Hogzilla**[10] é um Sistema de Detecção de Intrusão (IDS) de código aberto suportado por *softwares* livres, que fornecem detecção de anomalias na rede através do processamento em lote de grandes massas de dados. O Hogzilla emprega também ferramentas que permitem a visibilidade da rede.

4.7. Considerações Finais

As redes de acesso sem fio estão presentes em todos os ambientes. Os dados de acesso nessas redes também estão crescendo em volume, variedade e velocidade. Paralelamente, diversas pesquisas mostram que os dados dessas redes também apresentam grande valor, mas dependem da validação de sua veracidade. O valor dos dados de acesso em redes sem fio é crítico para o monitoramento, gerenciamento e controle da rede, mas também permite inferir conhecimentos sobre a infraestrutura, padrões de mobilidade, preferências e qualidade de experiência e dos serviços dos usuários. Assim, a análise dos dados gerados em uma rede de acesso sem fio de grande escala consiste em um problema de processamento grandes massas de dados. O paradigma de processamento de grandes massas de dados requer armazenamento, processamento e proteção eficientes em termos de latência de processamento, espaço de armazenamento em memória e banda de transmissão na rede. Esses requisitos representam também os principais desafios do processamento das grandes massas de dados em redes de acesso sem fio. Ademais, por se tratarem de fontes de dados ininterruptas que geram dados potencialmente infinitos e com atributos com grande variação, torna-se necessário a aplicação de ferramentas de processamento de grandes massas de dados em fluxo. Esse capítulo apresentou as principais ferramentas para o monitoramento de redes sem fio e aplicações de analíticas para redes. Destaca-se que metodologia como o aprendizado de máquina têm sido uma abordagem importante aplicada na classificação, na descoberta de padrões e na segurança das redes.

O capítulo discutiu os conceitos relacionados ao processamento em fluxo de dados e apresentou algoritmos de aprendizado de máquina incremental. Os algoritmos de

[7]Disponível em https://fibre.org.br/.

[8]Disponível em http://prestocloud-project.eu/.

[9]Disponível em http://metron.apache.org/.

[10]Disponivel em http://ids-hogzilla.org/.

aprendizado incremental tendem a assumir que a distribuição de probabilidade dos atributos considerados na formação do modelo não varia ao longo do tempo. Contudo, na ocorrência de uma variação, seja brusca ou amortecida no tempo, os algoritmos de aprendizado incremental tendem a falhar na extração de conhecimento dos dados entrantes. Um dos principais desafios para o processamento em fluxo de dados, então, é a detecção de mudanças nas estatísticas dos atributos dos dados entrantes e o desenvolvimento de algoritmos de aprendizado de máquina capazes de reagir a essas mudanças.

O capítulo analisou ainda as aplicações de última geração de aprendizado de máquina em redes sem fio e identificou as técnicas de aprendizado por agregado de classificadores, aprendizado profundo e aprendizado por reforço como promissoras para as próximas gerações de aplicações na extração do conhecimento no monitoramento de redes. Há ainda tendências de novas aplicações que adotam o aprendizado por transferência de conhecimento, em que o conhecimento extraído de um contexto é transferido para outra aplicação em execução em outro contexto. Dessa forma, a transferência de aprendizado evita treinar modelos de aprendizado a partir do zero e o processo de aprendizado em novos ambientes pode ser acelerado de modo em que o algoritmo de aprendizado de máquina pode ter um bom desempenho, mesmo com uma pequena quantidade de dados de treinamento. Portanto, transferir aprendizagem é fundamental para a implementação prática de modelos de aprendizado de máquina considerando o custo para treinamento sem conhecimento prévio. Usando o transferência de aprendizado, os operadores de rede podem resolver problemas novos, mas semelhantes a anteriores, de maneira mais econômica. No entanto, os efeitos negativos do conhecimento prévio sobre o desempenho do sistema necessitam maiores investigações. Ao se considerar as propostas em aprendizado profundo, identificam-se as tendências de adoção de técnicas mistas que combinam aprendizado profundo e por reforço. Além disso, são propostas redes de crença profunda (*deep belief networks*) que combinam diversas camadas de redes neurais, criando redes neurais ainda mais profundas, com mais camadas intermediárias profundas e diferentes ligações entre neurônios em cada camada.

Conclui-se que o processamento em fluxo de grandes massas de dados apresenta desafios significativos à aprendizagem profunda, incluindo grande escala, heterogeneidade, ruído na marcação dos rótulos e a distribuição não estacionária de atributos, entre outros. Assim, para realizar o potencial da analítica de grandes massas de dados em fluxo, há a necessidade de se enfrentar os desafios técnicos com propostas inovadoras e soluções transformadoras. Pesquisas em desafios impostos pela extração de conhecimento em grandes massas de dados em fluxo não são apenas oportunas, mas necessárias para diversos campos do conhecimento muito além de apenas a extração de conhecimento no gerenciamento de redes sem fio.

Referências

[Aaron et al., 2014] Aaron, B., Tamir, D. E., Rishe, N. D. e Kandel, A. (2014). Dynamic incremental k-means clustering. Em *2014 International Conference on Computational Science and Computational Intelligence*, volume 1, p. 308–313.

[Acer et al., 2015] Acer, U. G., Boran, A., Forlivesi, C., Liekens, W., Pérez-cruz, F. e Kawsar, F. (2015). Sensing wifi network for personal IoT analytics. Em *5th Internati-*

onal Conference on the Internet of Things, p. 104–111.

[Acer et al., 2016] Acer, U. G., Vanderhulst, G., Masshadi, A., Boran, A., Forlivesi, C., Scholl, P. M. e Kawsar, F. (2016). Capturing personal and crowd behavior with wi-fi analytics. Em *3rd International Workshop on Physical Analytics*, p. 43–48. ACM.

[Afanasyev et al., 2010] Afanasyev, M., Chen, T., Voelker, G. M. e Snoeren, A. C. (2010). Usage patterns in an urban wifi network. *IEEE/ACM Transactions on Networking (ToN)*, 18(5):1359–1372.

[Alessandrini et al., 2017] Alessandrini, A., Gioia, C., Sermi, F., Sofos, I., Tarchi, D. e Vespe, M. (2017). Wifi positioning and big data to monitor flows of people on a wide scale. Em *2017 European Navigation Conference (ENC)*, p. 322–328.

[Alsheikh et al., 2016] Alsheikh, M. A., Niyato, D., Lin, S., Tan, H.-P. e Han, Z. (2016). Mobile big data analytics using deep learning and apache spark. *IEEE Network*, 30(3):22–29.

[Andreoni Lopez et al., 2016] Andreoni Lopez, M., Lobato, A. G. P. e Duarte, O. C. M. B. (2016). A performance comparison of open-source stream processing platforms. Em *2016 IEEE Global Communications Conference (GLOBECOM)*, p. 1–6. IEEE.

[Andreoni Lopez et al., 2019] Andreoni Lopez, M., Mattos, D. M. F., Duarte, O. C. M. B. e Pujolle, G. (2019). A fast unsupervised preprocessing method for network monitoring. *Annals of Telecommunications*, 74.

[Andreoni Lopez et al., 2018] Andreoni Lopez, M., Sanz, I. J., Lobato, A. G. P., Mattos, D. M. F. e Duarte, O. C. M. B. (2018). Aprendizado de máquina em plataformas de processamento distribuído de fluxo: Análise e detecção de ameaças em tempo real. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC*, 2018:150–206.

[Armbrust et al., 2015] Armbrust, M., Xin, R. S., Lian, C., Huai, Y., Liu, D., Bradley, J. K., Meng, X., Kaftan, T., Franklin, M. J., Ghodsi, A. et al. (2015). Spark sql: Relational data processing in spark. Em *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, p. 1383–1394. ACM.

[Balbi et al., 2012] Balbi, H., Fernandes, N., Souza, F., Carrano, R., Albuquerque, C., Muchaluat-Saade, D. e Magalhaes, L. (2012). Centralized channel allocation algorithm for ieee 802.11 networks. Em *2012 Global Information Infrastructure and Networking Symposium (GIIS)*, p. 1–7.

[Blei e Smyth, 2017] Blei, D. M. e Smyth, P. (2017). Science and data science. *Proceedings of the National Academy of Sciences*, 114(33):8689–8692.

[Boutaba et al., 2018] Boutaba, R., Salahuddin, M. A., Limam, N., Ayoubi, S., Shahriar, N., Estrada-Solano, F. e Caicedo, O. M. (2018). A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, 9(1):16.

[Bozkurt et al., 2015] Bozkurt, S., Elibol, G., Gunal, S. e Yayan, U. (2015). A comparative study on machine learning algorithms for indoor positioning. Em *2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA)*, p. 1–8. IEEE.

[Carbone et al., 2015a] Carbone, P., Fóra, G., Ewen, S., Haridi, S. e Tzoumas, K. (2015a). Lightweight asynchronous snapshots for distributed dataflows. *arXiv preprint arXiv:1506.08603*.

[Carbone et al., 2015b] Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi†, S. e Tzoumas, K. (2015b). Apache flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 4(34):28–38.

[Chen e Lin, 2014] Chen, X. e Lin, X. (2014). Big data deep learning: Challenges and perspectives. *IEEE Access*, 2:514–525.

[Chen et al., 2018] Chen, Y., Guizani, M., Zhang, Y., Wang, L., Crespi, N., Lee, G. M. e Wu, T. (2018). When traffic flow prediction and wireless big data analytics meet. *IEEE Network*.

[Chen e Qiu, 2011] Chen, Z. e Qiu, R. C. (2011). Cooperative spectrum sensing using q-learning with experimental validation. Em *2011 Proceedings of IEEE Southeastcon*, p. 405–408.

[Chilipirea et al., 2016] Chilipirea, C., Petre, A., Dobre, C. e van Steen, M. (2016). Presumably simple: Monitoring crowds using wifi. Em *Proceedings of the 17th IEEE International Conference on Mobile Data Management (MDM)*, volume 1, p. 220–225.

[Chintapalli et al., 2016] Chintapalli, S., Dagit, D., Evans, B., Farivar, R., Graves, T., Holderbaugh, M., Liu, Z., Nusbaum, K., Patil, K., Peng, B. J. et al. (2016). Benchmarking streaming computation engines: Storm, flink and spark streaming. Em *2016 IEEE international parallel and distributed processing symposium workshops (IPDPSW)*, p. 1789–1792. IEEE.

[Cici et al., 2015] Cici, B., Gjoka, M., Markopoulou, A. e Butts, C. T. (2015). On the decomposition of cell phone activity patterns and their connection with urban ecology. Em *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '15, p. 317–326.

[Cisco, 2017] Cisco, V. N. I. (2017). Global mobile data traffic forecast update, 2016–2021 white paper. *Document ID*, 1454457600805266.

[Costa et al., 2012] Costa, L. H. M. K., de Amorim, M. D., Campista, M. E. M., Rubinstein, M., Florissi, P. e Duarte, O. C. M. B. (2012). Grandes Massas de Dados na Nuvem: Desafios e Técnicas para Inovação. Em *SBRC 2012 - Minicursos*.

[Dean e Ghemawat, 2008] Dean, J. e Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

[Deng, 2014] Deng, L. (2014). A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*, 3:e2.

[Deng e Yu, 2014] Deng, L. e Yu, D. (2014). Deep learning: Methods and applications. *Foundations and Trends in Signal Processing*, 7(3–4):197–387.

[Divgi e Chlebus, 2013] Divgi, G. e Chlebus, E. (2013). Characterization of user activity and traffic in a commercial nationwide wi-fi hotspot network: global and individual metrics. *Wireless Networks*, 19(7):1783–1805.

[Fan et al., 2016] Fan, Y., Chen, Y., Tung, K., Wu, K. e Chen, A. L. P. (2016). A framework for enabling user preference profiling through wi-fi logs. *IEEE Transactions on Knowledge and Data Engineering*, 28(3):592–603.

[Gaber, 2012] Gaber, M. M. (2012). Advances in data stream mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):79–85.

[Gama et al., 2014] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M. e Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):44.

[García et al., 2016] García, S., Ramírez-Gallego, S., Luengo, J., Benítez, J. M. e Herrera, F. (2016). Big data preprocessing: methods and prospects. *Big Data Analytics*, 1(1):9.

[Garofalakis et al., 2016] Garofalakis, M., Gehrke, J. e Rastogi, R. (2016). *Data Stream Management: Processing High-Speed Data Streams*. Springer.

[Godec et al., 2010] Godec, M., Leistner, C., Saffari, A. e Bischof, H. (2010). On-line random naive bayes for tracking. Em *2010 20th Int. Conference on Pattern Recognition*, p. 3545–3548.

[Goel et al., 2015] Goel, U., Wittie, M. P., Claffy, K. C. e Le, A. (2015). Survey of end-to-end mobile network measurement testbeds, tools, and services. *IEEE Communications Surveys & Tutorials*.

[Gonzalez et al., 2014] Gonzalez, J. E., Xin, R. S., Dave, A., Crankshaw, D., Franklin, M. J. e Stoica, I. (2014). GraphX: Graph processing in a distributed dataflow framework. Em *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, p. 599–613.

[Guo et al., 2018] Guo, J., Tan, Z.-H., Cho, S. H. e Zhang, G. (2018). Wireless personal communications: Machine learning for big data processing in mobile internet. *Wireless Personal Communications*, 102(3):2093–2098.

[Hadi et al., 2018] Hadi, M. S., Lawey, A. Q., El-Gorashi, T. E. e Elmirghani, J. M. (2018). Big data analytics for wireless and wired network design: A survey. *Computer Networks*, 132:180–199.

[Ham e Lee, 2014] Ham, Y. J. e Lee, H.-W. (2014). Big data preprocessing mechanism for analytics of mobile web log. *International Journal of Advances in Soft Computing & Its Applications*, 6(1).

[Han et al., 2011] Han, J., Pei, J. e Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

[Hofstede et al., 2014] Hofstede, R., Čeleda, P., Trammell, B., Drago, I., Sadre, R., Sperotto, A. e Pras, A. (2014). Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. *IEEE Communications Surveys & Tutorials*, 16(4):2037–2064.

[Hu et al., 2014] Hu, H., Wen, Y., Chua, T. e Li, X. (2014). Toward scalable systems for big data analytics: A technology tutorial. *IEEE Access*, 2:652–687.

[Huang et al., 2016] Huang, H., Cai, Y. e Yu, H. (2016). Distributed-neuron-network based machine learning on smart-gateway network towards real-time indoor data analytics. Em *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*, DATE '16, p. 720–725.

[Huang et al., 2014] Huang, W., Chen, Z., Dong, W., Li, H., Cao, B. e Cao, J. (2014). Mobile internet big data platform in china unicom. *Tsinghua Science and Technology*, 19(1):95–101.

[Iqbal e Soomro, 2015] Iqbal, M. H. e Soomro, T. R. (2015). Big data analysis: Apache storm perspective. *Int. journal of computer trends and technology*, 19(1):9–14.

[Isard et al., 2007] Isard, M., Budiu, M., Yu, Y., Birrell, A. e Fetterly, D. (2007). Dryad: distributed data-parallel programs from sequential building blocks. *ACM SIGOPS operating systems review*, 41(3):59–72.

[Jang et al., 2017] Jang, R., Cho, D., Noh, Y. e Nyang, D. (2017). Rflow+: An sdn-based wlan monitoring and management framework. Em *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, p. 1–9.

[Jiang et al., 2017] Jiang, C., Zhang, H., Ren, Y., Han, Z., Chen, K.-C. e Hanzo, L. (2017). Machine learning paradigms for next-generation wireless networks. *IEEE Wireless Communications*, 24(2):98–105.

[Jordaney et al., 2017] Jordaney, R., Sharad, K., Dash, S. K., Wang, Z., Papini, D., Nouretdinov, I. e Cavallaro, L. (2017). Transcend: Detecting concept drift in malware classification models. Em *PROCEEDINGS OF THE 26TH USENIX SECURITY SYMPOSIUM*, p. 625–642. USENIX Association.

[Kim et al., 2010] Kim, S. K., McMahon, P. L. e Olukotun, K. (2010). A large-scale architecture for restricted boltzmann machines. Em *18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, p. 201–208.

[Kwon et al., 2017] Kwon, D., Kim, H., Kim, J., Suh, S. C., Kim, I. e Kim, K. J. (2017). A survey of deep learning-based network anomaly detection. *Cluster Computing*.

[Lee et al., 2005] Lee, G. M., Liu, H., Yoon, Y. e Zhang, Y. (2005). Improving sketch reconstruction accuracy using linear least squares method. Em *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, IMC '05, p. 24–24, Berkeley, CA, USA. USENIX Association.

[Leung e Kim, 2003] Leung, K. K. e Kim, B. J. (2003). Frequency assignment for IEEE 802.11 wireless networks. Em *2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No.03CH37484)*, volume 3, p. 1422–1426 Vol.3.

[Li et al., 2013] Li, B., Springer, J., Bebis, G. e Hadi Gunes, M. (2013). Review: A survey of network flow applications. *Journal of Network and Computer Applications*, 36(2):567–581.

[Li, 2018] Li, Y. (2018). Deep Reinforcement Learning. *arXiv e-prints*, p. arXiv:1810.06339.

[Liebchen et al., 2007] Liebchen, G., Twala, B., Shepperd, M., Cartwright, M. e Stephens, M. (2007). Filtering, robust filtering, polishing: Techniques for addressing quality in software data. Em *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, p. 99–106. IEEE.

[Lin et al., 2017] Lin, F. Y. S., Wen, Y. F., Fang, L. W. e Hsiao, C. H. (2017). Resource allocation and multisession routing algorithms in coordinated multipoint wireless communication networks. *IEEE Systems Journal*, PP(99):1–12.

[Liu e Yoo, 2017] Liu, Y. e Yoo, S. (2017). Dynamic resource allocation using reinforcement learning for lte-u and wifi in the unlicensed spectrum. Em *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, p. 471–475.

[Lobato et al., 2018] Lobato, A. G. P., Andreoni Lopez, M., Sanz, I. J., Cardenas, A. A., Duarte, O. C. M. B. e Pujolle, G. (2018). An adaptive real-time architecture for zero-day threat detection. Em *2018 IEEE International Conference on Communications (ICC)*, p. 1–6.

[Low et al., 2012] Low, Y., Bickson, D., Gonzalez, J., Guestrin, C., Kyrola, A. e Hellerstein, J. M. (2012). Distributed graphlab: a framework for machine learning and data mining in the cloud. *Proceedings of the VLDB Endowment*, 5(8):716–727.

[Low et al., 2014] Low, Y., Gonzalez, J. E., Kyrola, A., Bickson, D., Guestrin, C. E. e Hellerstein, J. (2014). Graphlab: A new framework for parallel machine learning. *arXiv preprint arXiv:1408.2041*.

[Luiz et al., 2015] Luiz, T. A., Freitas, A. R. e Guimarães, F. G. (2015). A new perspective on channel allocation in wlan: Considering the total marginal utility of the connections for the users. Em *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, p. 879–886, New York, NY, USA. ACM.

[Magalhães e Mattos, 2018] Magalhães, L. C. S. e Mattos, D. M. F. (2018). Caracterização do uso de uma rede sem fio de grande porte distribuída por uma ampla Área. *XVII*

Workshop em Desempenho de Sistemas Computacionais e de Comunicação (WPerformance - CSBC 2018), 17(1/2018).

[Malewicz et al., 2010] Malewicz, G., Austern, M. H., Bik, A. J., Dehnert, J. C., Horn, I., Leiser, N. e Czajkowski, G. (2010). Pregel: a system for large-scale graph processing. Em *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, p. 135–146. ACM.

[Mattos et al., 2018] Mattos, D. M. F., Velloso, P. B. e Duarte, O. C. M. B. (2018). Uma infraestrutura Ágil e efetiva de virtualização de funções de rede para a internet das coisas. Em *XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'2018*.

[Maturi et al., 2017] Maturi, F., Gringoli, F. e Cigno, R. L. (2017). A dynamic and autonomous channel selection strategy for interference avoidance in 802.11. Em *2017 13th Annual Conference on Wireless On-demand Network Systems and Services(WONS)*, p. 1–8.

[Meng et al., 2016] Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S. et al. (2016). Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, 17(1):1235–1241.

[Monteiro et al., 2016] Monteiro, A., Souto, E., Pazzi, R. e Kiljander, J. (2016). Atribuição dinâmica de canais em redes sem fio não coordenadas ieee 802.11, baseada em fatores de sobreposição e intensidade de sinal. Em *XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'2016*.

[Moura et al., 2019] Moura, H. D., Macedo, D. F. e Vieira, M. A. M. (2019). Automatic quality of experience management for wlan networks using multi-armed bandit. Em *IFIP / IEEE International Symposium on Integrated Network Management*, p. 1–10. IEEE/IFIP.

[Qiu et al., 2016] Qiu, J., Wu, Q., Ding, G., Xu, Y. e Feng, S. (2016). A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing*, 2016(1):67.

[Shin et al., 2004] Shin, M., Mishra, A. e Arbaugh, W. A. (2004). Improving the latency of 802.11 hand-offs using neighbor graphs. Em *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services*, MobiSys '04, p. 70–83, New York, NY, USA. ACM.

[Song et al., 2010] Song, C., Qu, Z., Blumm, N. e Barabási, A.-L. (2010). Limits of predictability in human mobility. *Science*, 327(5968):1018–1021.

[Sun et al., 2018] Sun, Y., Peng, M., Zhou, Y., Huang, Y. e Mao, S. (2018). Application of machine learning in wireless networks: Key techniques and open issues. Relatório técnico, arXiv preprint arXiv:1809.08707. Online. `https://arxiv.org/abs/1809.08707`. Acessado em 22/03/2019.

[Sze et al., 2017] Sze, V., Chen, Y., Yang, T. e Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329.

[Tabrizi et al., 2011] Tabrizi, H., Farhadi, G. e Cioffi, J. (2011). A learning-based network selection method in heterogeneous wireless systems. Em *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, p. 1–5.

[Tannahill e Jamshidi, 2014] Tannahill, B. K. e Jamshidi, M. (2014). System of systems and big data analytics–bridging the gap. *Computers & Electrical Engineering*, 40(1):2–15.

[Tesauro, 2007] Tesauro, G. (2007). Reinforcement learning in autonomic computing: A manifesto and case studies. *IEEE Internet Computing*, 11(1):22–30.

[Toch et al., 2018] Toch, E., Lerner, B., Ben-Zion, E. e Ben-Gal, I. (2018). Analyzing large-scale human mobility data: a survey of machine learning methods and applications. *Knowledge and Information Systems*, p. 1–23.

[Tsai et al., 2015] Tsai, C.-W., Lai, C.-F., Chao, H.-C. e Vasilakos, A. V. (2015). Big data analytics: a survey. *Journal of Big Data*, 2(1):21.

[Turgut et al., 2019] Turgut, Z., Üstebay, S., Zeynep Gürkaş Aydın, G. e Sertbaş, A. (2019). Deep learning in indoor localization using WiFi. Em Boyaci, A., Ekti, A. R., Aydin, M. A. e Yarkan, S., editors, *International Telecommunications Conference*, p. 101–110, Singapore. Springer.

[Wang et al., 2018] Wang, F., Gong, W., Liu, J. e Wu, K. (2018). Channel selective activity recognition with WiFi: A deep learning approach exploring wideband information. *IEEE Transactions on Network Science and Engineering*, p. 1–1.

[Wang et al., 2013] Wang, S., Minku, L. L., Ghezzi, D., Caltabiano, D., Tino, P. e Yao, X. (2013). Concept drift detection for online class imbalance learning. Em *The 2013 Int. Joint Conference on Neural Networks (IJCNN)*, p. 1–10.

[Wang et al., 2018] Wang, T., Yang, Q., Tan, K., Zhang, J., Liew, S. C. e Zhang, S. (2018). Dcap: Improving the capacity of wifi networks with distributed cooperative access points. *IEEE Transactions on Mobile Computing*, 17(2):320–333.

[Wang et al., 2017] Wang, X., Gao, L. e Mao, S. (2017). BiLoc: Bi-Modal Deep Learning for Indoor Localization With Commodity 5GHz WiFi. *IEEE Access*, 5:4209–4220.

[Wang et al., 2018] Wang, X., Wang, X. e Mao, S. (2018). RF sensing in the internet of things: A general deep learning framework. *IEEE Communications Magazine*, 56(9):62–67.

[Warneke e Kao, 2009] Warneke, D. e Kao, O. (2009). Nephele: efficient parallel data processing in the cloud. Em *Proceedings of the 2nd workshop on many-task computing on grids and supercomputers*, p. 8. ACM.

[Xin et al., 2013] Xin, R. S., Rosen, J., Zaharia, M., Franklin, M. J., Shenker, S. e Stoica, I. (2013). Shark: SQL and rich analytics at scale. Em *Proceedings of the 2013 ACM SIGMOD International Conference on Management of data*, p. 13–24. ACM.

[Xu et al., 2017] Xu, G., Gao, S., Daneshmand, M., Wang, C. e Liu, Y. (2017). A survey for mobility big data analytics for geolocation prediction. *IEEE Wireless Communications*, 24(1):111–119.

[Zaharia et al., 2012a] Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S. e Stoica, I. (2012a). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. Em *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, p. 2–2. USENIX Association.

[Zaharia et al., 2013] Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S. e Stoica, I. (2013). Discretized streams: Fault-tolerant streaming computation at scale. Em *XXIV ACM Symposium on Operating Systems Principles*, p. 423–438. ACM.

[Zaharia et al., 2012b] Zaharia, M., Das, T., Li, H., Shenker, S. e Stoica, I. (2012b). Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters. *Proceedings of the 4th USENIX conference on Hot Topics in Cloud Ccomputing*, p. 10–10.

[Zeng et al., 2015] Zeng, Y., Pathak, P. H. e Mohapatra, P. (2015). Analyzing shopper's behavior through wifi signals. Em *Proceedings of the 2Nd Workshop on Physical Analytics*, WPA '15, p. 13–18.

[Zeng et al., 2016] Zeng, Y., Pathak, P. H. e Mohapatra, P. (2016). Wiwho: Wifi-based person identification in smart spaces. Em *Proceedings of the 15th International Conference on Information Processing in Sensor Networks*, IPSN '16, p. 4:1–4:12.

[Zhang et al., 2016] Zhang, X., Wang, C., Li, Z., Zhu, J., Shi, W. e Wang, Q. (2016). Exploring the sequential usage patterns of mobile internet services based on markov models. *Electronic Commerce Research and Applications*, 17:1 – 11.