Survey: Análise de Streams de Dados

Miguel Sandim e Paula Fortuna

Faculdade de Engenharia da Universidade do Porto,
{miguel.sandim, paula.fortuna}@fe.up.pt

1 Introdução

A área de Machine Learning tem adquirido maior visibilidade nos últimos anos. Nesta área o método mais comum de análise pode ser designado de *batch learning* [1], que é um método de aprendizagem em que existe um *dataset* disponível na totalidade durante todo o processo de análise e, depois de este ser processado, é produzido um modelo.

A ideia por trás desta abordagem é a de que existe uma população, que segue uma distribuição probabilística constante no tempo, da qual o conjunto de dados é uma amostra. Note-se que este tipo de análise é também recorrente em estatística clássica. Contudo, esta abordagem apresenta algumas desvantagens. Em primeiro lugar, está sujeita a uma grande variância [1], dado que é tida em conta apenas uma amostra para construção do modelo. Outra desvantagem, que pode ser enumerada, é a de que o modelo de dados construído permanece estático ao longo do tempo.

Por outro lado, com a massificação da utilização da Internet e dos Sistemas de Informação, o volume de dados disponível para análise é cada vez maior e é possível aceder a dados de forma contínua no tempo. Por exemplo, o Twitter [2] disponibiliza uma API para *streaming* que pode ser usada em *real time* para extrair dados da rede social.

Neste contexto, e para dar resposta às limitações das abordagens anteriores, a área de Análise de *Streams* de Dados tem crescido e adquirido maior relevância.

Neste *survey* pretende-se aprofundar esta área, nomeadamente através dos seguintes objetivos:

- Perceber mais em detalhe de que forma a Análise de *Streams* de Dados se distingue das abordagens mais comuns na área de *Machine Learning* e *Data Mining*;
- Analisar quais os seus principais algoritmos e a que problemas estes dão resposta;
- Compreender como avaliar algoritmos no caso de Análises de *Streams* de Dados.
- Descrever a evolução desta área de investigação nos últimos anos.
- Analisar as questões de investigação mais iminentes nesta área.

Será dedicada uma secção a cada um dos temas enumerados.

2 Data Streams vs. Abordagens Mais Comuns em Análises de Dados

Antes de se comparar a análise de *Data Streams* com as abordagens mais comuns em análises de dados, é importante definir o que se entende por este tipo de dados. *Data*

streams são processos estocásticos em que os eventos ocorrem de forma independente entre si, de forma contínua [1] e em que os dados a operar não estão em disco, disponíveis para *random access*, mas que chegam através de fluxos de dados [3].

Algumas das principais características, que distinguem *data streams* podem ser enumeradas [1] [3]:

- Os elementos da data stream chegam *on-line*.
- O sistema não contola a ordem de chegada dos elementos, em uma ou várias *data streams*.
- Uma *data stream* é ilimitada em número de elementos.
- Quando um elemento de uma *data stream* é processado, é descartado ou guardado. Contudo, não pode ser recuperado facilmente, a não ser que esteja explicitamente guardado em memória, que é pequena, comparativamente ao tamanho da *data stream*.

Na literatura distingue-se *Data Base Management Systems* (DBMS) de *Data Streams Management Systems* (DSMS) [1] e [3]. DBMS referem-se às abordagens mais comuns, que usam o modelo relacional convencional, enquanto as DSMS se referem à utilização de *Data Strems*. Apresentam-se na tabela 1 as principais diferenças entre as duas [1].

Tabela 1. Diferenças entre DBMS e DSMS

Data Base Management Systems	**Data Streams Management Systems**
Relações Persistentes	Streams Transientes
Queries one-time	Queries contínuas
Acesso Random	Acesso Sequencial
Acesso determinado pela arquitectura da base de dados	Características imprevisíveis
Múltiplas passagens nos dados	Passagem única nos dados
Tempo de processamento ilimitado	Tempo de processamento restrito
Memória ilimitada	Memória restrita
Resultado preciso	Resultado aproximado
Sistema Centralizado	Sistema Distribuído

No próximo capítulo apresentam-se os principais algoritmos utilizados nesta área.

3 Algoritmos em *Data Streams*

As diferenças entre *data streams* e as abordagens mais comuns em análises de dados, sistematizadas no capítulo anterior, repercutem-se nas estratégias utilizadas para resolver problemas e formular algoritmos. Apresentam-se na imagem 1 as principais categorias de algoritmos em *data streams* e alguns exemplos principais [1] [4].

Nos próximos subcapítulos serão definidas estas categorias, bem como alguns dos seus princípios.

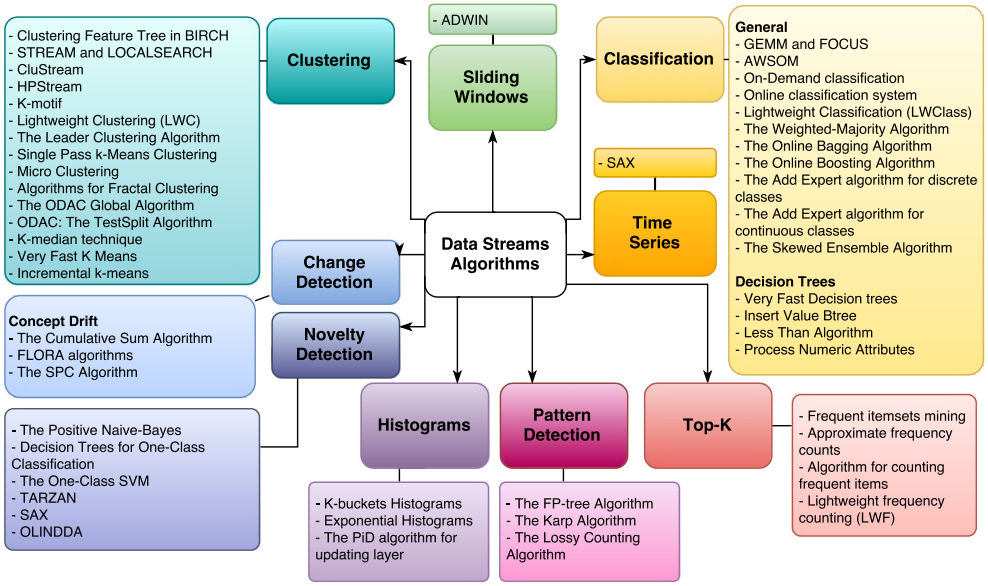

Figura 1. Categorias e exemplos de algoritmos em *data streams*.

3.1 *Clustering*

A área de *clustering* em *data streams* tem tido um forte enfoque nos últimos anos [5].

Identificam-se quatro requisitos particulares que têm de ser verificados nos algoritmos em *clustering* em *data streams* [6]: representação compacta; processamento rápido e incremental de novas instâncias de dados; monitorização de mudanças nos *clusters*; e, por fim, limpeza e identificação de *outliers*. Outros autores referem ainda que os algoritmos em *data stream clustering* podem dividir-se em dois tipos [5]: algoritmos clássicos e novos algoritmos.

3.1.1 Algoritmos clássicos em *data stream clustering* - Os primeiros algoritmos a surgir nesta área focavam-se em construir uma estrutura de dados de forma semelhante aos métodos tradicionais em *clustering*. Por exemplo, um dos primeiros algoritmos desenvolvidos foi o Birch (Zhang e colegas em 1996) [5]. Na sequência deste algoritmo seguiram-se outros, nomeadamente, LocalSearch (Guha e colegas em 2003), Stream (O'Callaghan e colegas em 2002), CluStream (Aggarwal e colegas em 2003), M-Birch (Ling e colegas 2007), HPstream (Aggarwal e colegas 2004), CD-Stream (Sun e colegas 2004), A-Clustream (Zhu e colegas 2006), CLUSMD (Ni e colegas) e, por fim, algoritmos baseados em densidade de probabilidades (Song e Wang 2005).

3.1.2 Novos algoritmos em *data stream clustering* - Com o avançar da investigação esta focou-se mais em dois aspetos [5]: em primeiro, *clustering* em dados

com distribuições irregulares; em segundo, em como detetar *outliers* e excluir a interferência de ruído nos dados. Destacamse: *sub-space projection technique* (Aggarwal e colegas em 2005) para resolver o problema de *dimensions disaster*, UMicro para *clustering* em *data streams* incertas (Aggarwal e Yu em 2008), estratégias com base em probabilidades (Wang e colegas em 2009), *wavelet synopsis* (Chen e Shi em 2010) e *feasible data stream clustering* (Kavitha e Punithavalli em 2010).

3.2 Change Detection

Change detection refere-se a um conjunto de algoritmos que combinam tanto robustez a ruído, como sensibilidade a mudança de conceito [1]. Um exemplo de algoritmo em *change detection* é o Cumulative Sum (CUSUM) de Page em 1954 [1]. Contudo, apesar de ser uma temática em estudo desde o início da área de *data streaming*, em artigos mais recentes a expressão *change detection* não é tão referida, sendo mais comum encontrar-se artigos dedicados a *concept drifting*.

3.2.1 ***Concept drift*** - Significa que o conceito sobre o qual estão a ser recolhidos dados pode mudar e, nesse caso, observações antigas que refletiam a natureza do comportamento passado, tornam-se obsoletas, sendo necessário esquecer essa informação. Os métodos para detetar *concept drift* podem seguir duas abordagens [1]:

- Monitorizar a evolução de indicadores (e. g. Klinkenberg e Renz em 1998) [1].
- Monitorizar a distribuição de duas janelas de tempo distintas, utilizando uma janela de referência para sumarizar informação sobre o passado e uma janela para exemplos mais recentes (Kifer e colegas em 2004) [1].

A maioria das abordagens segue a primeira perspetiva, salientando-se os algoritmos FLORA. Contudo, atualmente outros algoritmos, como as VFDTc (Gama e colegas em 2006) conseguem lidar com *concept drifting* ao monitorizar continuamente as diferenças entre as distribuições de duas classes [1].

Outra abordagem seguida, pode ainda ser a de utilizar métodos de adaptação. Por fim, é ainda de salientar que tem também sido utilizado Statistical Process Control (SPC), como algoritmo para detetar concept drifting.

3.3 *Novelty Detection*

Novelty detection é o processo de identificação de informação desconhecida até ao momento e distingue-se de *concept drifting*, dado que não se trata apenas de detetar mudança, mas sim novidade. Esta capacidade de identificar novidade é central nas tarefas de machine learning contínuas.

Na maioria das abordagens em *novelty detection*, o objetivo é definir quando um novo exemplo recolhido está de acordo com o previsto no modelo ou não. Assim, uma vez que, por definição, não existem exemplos de conceitos desconhecidos, em *novelty detection* o treino é feito apenas com exemplos positivos, de onde advém o termo *one-class classification* para estes algoritmos. Alguns dos algoritmos utilizados, são

The Positive Naive-Bayes (Denis e colegas em 2005), Decision Trees para One-Class Classification (Denis e colegas em 2005) e The One-Class SVM (Liu e colegas em 2003).

3.4 Classificação

Os problemas de classificação, ou seja, de atribuir uma classe a uma nova instância, são dos mais estudados em Machine Learning e têm também sido adaptados ao contexto de *data streams*. Neste sentido, vários algoritmos têm sido desenvolvidos [7], o que se apresenta de seguida.

Relativamente a árvores de decisão, os principais algoritmos procuram lidar com *concept drifting* [7]. Very Fast Decision Trees (VFDT) (Domingos e colegas em 2000) é um dos algoritmos mais conhecidos, em que os exemplos para treino não são mantidos, aos quais foram sucedendo outros algoritmos para ultrapassarem algumas das suas limitações, nomeadamente CVFDT (Hulten e colegas em 2001) que já lida com *concept drift* e VFDTc (Gama e colegas em 2003) que permite atributos numéricos e não apenas categóricos. Surgiram também o IADEM (Ramos-Jimenez e colegas em 2006) e o IADEMc (del Campo Avila e colegas em 2006) também baseados no Hoeffding *bound*.

Uma desvantagem de algoritmos com árvores é que estas têm uma estrutura rígida que pode necessitar de uma reestruturação completa para lidar com *concept drifting*.

Existem ainda outros algoritmos que foram adaptados a *data streams*: Support Vector Machine (SVM), nomeadamente, usando uma aproximação da solução ótima com "MEB - Minimum Enclosing Balls" (Tsang e colegas em 2006), ou ainda LASVM (Bordes e colegas em 2005); também *Rule-based system*, com o algoritmo FACIL ou o algoritmo de Gama e Kosina; Naive Bayes, que usam classificação com base em probabilidade condicionada, como o algoritmo PiD: "Partition Incremental Discretization" (Gama e Pinto em 2006); e, por fim, KNN (por exemplo, Law e colegas em 2005), que podem ser facilmente adaptados a *streams*, uma vez que podem ser guardados exemplos representativos, esquecendo-se exemplos mais antigos, ou então aglomerando vizinhos.

3.5 *Sliding Windows*

Em *data stream mining*, muitas vezes não se tem interesse em calcular parâmetros ou *queries* para todo o conjunto de dados conhecidos, mas apenas para um subconjunto de dados mais recente. Neste caso existe uma estrutura de tamanho fixa (*sliding window*), onde se guarda o conjunto de dados. Assim, quando um novo dado é recolhido, o elemento mais antigo da estrutura é removido [1].

Existem dois tipos de *sliding windows* segundo Babcock e colegas [1]:

– **Baseadas na sequência** - O tamanho da janela é definida em termos de número de observações (espaço).

– **Baseadas em *timestamps*** - O tamanho da janela é definido em termos de duração (tempo). Assim, os elementos da janela encontram-se dentro de um intervalo de tempo.

Depois de se constituir a janela, as métricas pretendidas podem ser calculadas sobre todos os seus elementos. Outra abordagem possível é manter estatísticas aproximadas sobre a janela, sem guardar todos os elementos.

Sliding windows é uma técnica utilizada em diversos algoritmos, inclusive em área distintas de *data streams*. O ADWIN é um exemplo de algoritmo de *sliding windows* em que o tamanho da janela é variável, o que permite poupar espaço.

3.6 *Top-k*

É um dos problemas mais comuns em *data streams* e consiste em encontrar os elementos que ocorrem mais frequentemente num conjunto de dados. Este problema ocorre quando existe um elevado volume de dados imprevisível e manter um contador para cada item é demasiado custoso em termos espaciais [8].

Uma solução pode ser a de manter apenas uma amostra de elementos, e um contador para cada elemento. Neste âmbito aparecem os algoritmos *space-saving* (Later e colegas em 2005) [1]. Estes algoritmos são eficientes para alfabetos vastos e permitem que os elementos populares evoluam com o tempo, dado que tendem a dar mais importância a observações mais recentes. Outros algoritmos utilizados são Frequent itemsets mining, Approximate frequency counts, Algorithm for counting frequent items e Lightweight frequency counting (LWF).

3.7 Histogramas

Um histograma é um conjunto de intervalos numéricos não sobrepostos, sendo cada intervalo definido por limites e uma frequência de ocorrências [1]. Contudo, quando os dados chegam através de *data streams*, construir um histograma pode tornar-se problemático porque o domínio dos valores não é conhecido à partida.

Uma metodologia utilizada era a de seguir a regra de Sturges. Contudo, esta não funciona para um número superior a cerca de 200 ocorrências [1] e, portanto, dado que existem cada vez maiores volumes de dados para análise, outras soluções têm de ser desenvolvidas. Para isso surgem análises de dados exploratórias, em que os histogramas são usados iterativamente, ou seja, vão sendo construídos utilizando-se diferentes valores e intervalos escolhendo-se os que melhor servirem os propósitos do problema.

Os principais algoritmos utilizados são K-buckets Histograms, Exponential Histograms e The Partition Incremental Discretization Algorithm.

3.8 P*attern Detection*

O problema de encontrar padrões nos dados e conjuntos de itens frequentes, mais conhecido através do exemplo do problema de descobrir quais os itens comprados

no mesmo carrinho de compras, apresenta também particularidades no caso de *data streams*. Para além das restrições que os algoritmos em *data streams* apresentam, já descritas previamente, salienta-se que, neste problema em particular, a maior dificuldade advém do facto de itens pouco frequentes no passado poderem tornar-se frequentes no presente. Os principais algoritmos para este problema são FP-tree, Karp e Lossy Counting [1].

3.9 *Time Series*

Time series são conjuntos de dados de sequências de valores consecutivos retirados com um certo intervalo de tempo. Este tema está amplamente estudado na área da estatística e processamento de sinal. Contudo, ao mesmo tempo, são um caso particular de *data streams*. Muitos dos padrões em *time series* podem ser referidos em termos de trend (padrão que se altera ao longo do tempo) ou sazonalidade (padrão que se repete em intervalos sistemáticos).

Uma das principais abordagens em *time series* é SAX (Symbolic Approximation) e consiste em converter uma sequência de números, com domínio real, numa sequência de símbolos.

Depois de apresentadas as principais categorias de algoritmos em *data streams* apresentam-se as especificidades da avaliação de algoritmos neste contexto.

4 Avaliação de Algoritmos em *data streams*

Existem alguns indicadores para avaliação de algoritmos que podem ser usados tanto em *data streams* como em *batch learning*, nomeadamente: Accuracy (ACC), Balanced Accuracy (BER), Area Under the ROC curve (AUC), Kappa Statistic (K), Kappa Plus Statistic (K+) [7].

Contudo, algumas das metodologias mais comuns para avaliar algoritmos, baseadas em amostragem, não são adequadas em *data streams* [1]. Por exemplo, o método de *cross-validation* é apropriado se os *datasets* tiverem um tamanho restrito, com uma distribuição estática e assumindo que os exemplos são independentes entre si, contudo, estas propriedades não se verificam em *data streams*.

No contexto de *data streams* surgem outras técnicas de avaliação de algoritmos. Estas técnicas são [1]:

- ***Holdout evaluation*** - aplicar o modelo atual a um conjunto de teste independente, podendo este conjunto ser atualizado em intervalos de tempo regulares.
- ***Predictive Sequential*: prequência** - referida por Dawid em 1984, onde cada item é usado para treinar e testar o modelo caso a sua classe seja conhecida. O erro-prequencial é calculado com base na soma acumulada da função de erro entre o valor predito e observado.

Relativamente a estas duas opções, elas relacionam-se entre si. Por um lado, o estimador prequencial é pessimista, porque para os mesmos dados reporta maior erro. Neste sentido, esta última técnica foi melhorada, tendo sido incorporados mecanismos

de esquecimento para melhorar os resultados. As soluções desenvolvidas utilizam *sliding windows* ou um fator de ponderação que valorize a informação mais recente.

Por outro lado, em ambos os casos, ou seja, tanto com *sliding windows* como com ponderação, os valores do erro na prequência acabam por convergir para os mesmos resultados que "retirar um conjunto de teste independente".

4.1 Avaliação de algoritmos quando ocorre *concept drifting*

Um problema adicional coloca-se quando a *data stream* é não estática, ou seja, quando for provável que venha a ocorrer *concept drifting*. Existem alguns métodos para detetar mudança em *data streams* nestes casos [1]: probabilidade de falso alarme, probabilidade de alarme verdadeiro e *Delay* na deteção. Neste sentido, vários testes foram apresentados na literatura, sendo o Page-Hinkley o mais conhecido [1]. Este consiste numa técnica de análise sequencial tipicamente usada para monitorizar deteção de mudança.

5 Tendências na Literatura

De modo a se entenderem as tendências na literatura neste tema e sua evolução nos últimos anos, foi feita uma pesquisa por publicações nas plataforma Scopus e ACM Library recorrendo à *keyword* "data stream" entre 2005 e 2015 e nas áreas "Computer Science", "Engineering", "Mathematics", "Social Sciences", "Decision Sciences", "Business, Management and Accounting", "Economics, Econometrics and Finance" (no caso do Scopus). As *keywords* referidas nestas publicações foram combinadas e a sua frequência nestas foi calculada, estando os resultados sumarisados na tabela 2.

Numa primeira análise foi possível observar um crescente número de publicações em cada ano recolhido, sendo mais notório o crescimento entre 2005 e 2010 (de 784 para 1195 publicações no Scopus e de 373 para 510 publicações na ACM Library). Em 2005 as *keywords* mais utilizadas nas publicações no Scopus não estavam diretamente relacionadas com *data streaming* mas sim com as áreas de "Data Mining" e Bases de Dados, sendo que *keywords* mais teóricas e gerais, como "algorithms" (a mais frequente neste ano), "mathematical models", "data processing" e "database systems" eram as mais populares. No caso da ACM Library também se observa este fenómeno ("approximation algorithms", "statistical inference"), contudo observam-se também algumas *keywords* mais relacionadas com a área ("change detection", "continuous query"). Em anos mais recentes no Scopus e na ACM Library verificou-se um aumento na utilização de palavras-chave mais específicas e relacionadas com a área e as suas aplicações práticas, como "data communication systems" (a mais frequente em 2010 e 2015 no Scopus), "sliding window", "clustering" e "concept drift".

Relativamente a novas entradas no top de 2010 e 2015, as palavras-chave nestas condições relacionam-se mais com aplicações práticas na área (e.g., "sensors"/"sensor networks"/"sensor data", "anomaly detection" em 2010 e "big data", "social networking (online)", "learning systems", "cloud computing", "complex event processing" em 2015) e abordagens específicas na área (e.g., "clustering, "sliding window", "concept drift", "classification" em 2010).

Tabela 2. Listagem das dez *keywords* mais populares em publicações obtidas com a *keyword* "data stream" no Scopus e ACM Library nos 2005, 2010 e 2015, acompanhadas da percentagem de frequência nas publicações de cada ano. A cor da *keyword* indica se a mesma foi mais (verde) ou menos (vermelho) frequente no ano atual do que no ano anterior.

Scopus

2005	%	2010	%	2015	%
algorithms	16	data communication systems	40	data communication systems	27
data mining	13	hydraulics	24	data mining	22
data processing	12	data mining	21	big data	15
data reduction	12	algorithms	15	data handling	12
database systems	11	data processing	7	social networking (online)	7
data structures	9	sensors	6	clustering	6
data acquisition	8	clustering	6	learning systems	5
mathematical models	8	sliding window	6	classification (of information)	5
data communication systems	8	optimization	5	algorithms	4
computer simulation	8	data handling	5	digital storage	4
784 publicações		1195 publicações		1292 publicações	

ACM Library

2005	%	2010	%	2015	%
data mining	1.3	data mining	3.7	big data	5.7
single data stream architectures	1.3	clustering	3.3	concept drift	3.4
approximation algorithms	1.1	sliding window	2.0	classification	3.2
query processing	1.1	anomaly detection	1.8	data mining	2.5
statistical inference	1.1	classification	1.8	machine learning	2.3
change detection	0.8	concept drift	1.6	cloud computing	2.3
compression	0.8	sensor networks	1.6	clustering	2.1
continuous query	0.8	continuous queries	1.4	complex event processing	2.1
indexing	0.8	fpga	1.2	incremental learning	2.1
information retrieval	0.8	sensor data	1.2	continuous queries	1.6
373 publicações		510 publicações		439 publicações	

A análise das *keywords* nesta recolha permite concluir que:

- A frequência alta da palavra-chave "data communication systems" e o aparecimento de palavras-chave relativas a redes de sensores no top de 2010 de ambas as fontes poderá indicar que dispositivos tais como sensores (que registam e enviam informações continuamente) são das fontes de informação mais utilizadas nos últimos anos.
- Mais recentemente também é possível observar que *social networks* poderão estar-se a tornar numa fonte de dados importante, dado ser a 5ª palavra-chave mais frequente em 2015 no Scopus. Este fenómeno pode ser explicado pelo facto de nos últimos anos redes sociais como o Twitter e o Facebook terem verificado grandes aumentos de informação publicada por unidade de tempo (o que torna difícil a análise usando *batch learning*), assim como o aparecimento de APIs neste tipo de redes que permite de forma fácil a obtenção de dados em *streaming*.
- As abordagens relacionadas com *concept drifting*, *clustering* e classificação parecem revelar-se como as mais populares na literatura.

6 Questões em aberto

Depois de analisados os tópicos mais relevantes e sobre os quais tem incidido a maior parte das publicações nos últimos anos, destacam-se algumas questões de investigação atuais que poderão ser analisadas no futuro desta área. Estas questões são:

- Encontrar métodos que permitam diminuir a dimensionalidade, dado que esta afecta eficiência e taxa de acerto dos algoritmos [5] [9].
- Diminuir a complexidade computacional associada ao processo de limpeza de dados em streaming [5].
- Dada a falta de informação para algumas análises estatísticas, é necessário desenvolver métodos para lidar com incerteza em data streams [5].
- Melhorar a capacidade de predição em séries temporais [5].
- Introduzir o método de *subspacing* em *clustering* [5].
- Aplicar os algoritmos a domínios particulares e desenvolvê-los nesse contexto [5].
- Analisar questões de segurança em *data streams* [10].
- Investigar a utilização de *multithreading* e processamento paralelo para optimizar operações [11].
- Maior avaliação dos algoritmos em *datasets* reais [9].
- Reduzir o número de parâmetros necessários nos diversos algoritmos sobretudo os que têm um número mais elevado de parâmetros para funcionar (e. g. MR-Stream necessita de sete parâmetros) [9].
- Melhorar a performance dos algoritmos em termos de espaço, tempo, capacidade de acerto e também no funcionamento de soluções distribuídas [12].

Para além destas questões, também se tem verificado uma maior aplicação de análise de *data streams*, o que se espera que continue a ocorrer no futuro, nomeadamente: análise de espaços públicos [9], *smart cities* [13], serviços mobile de *internet* [14], *web clickstream* [15], eficiência energética [16], deteção de *bots* [17] [18], *social media* [19] [20], dados de sensores [21], análises de *logs* [22], extração de opiniões de *reviews* de produtos [23], predição da lealdade de clientes [24] e próximo ponto de interesse de pessoas [25].

7 Conclusão

O presente *survey* permitiu uma análise do panorama geral da área de análise de *Data Streams*, tendo esta sido distinguida das abordagens mais comuns em *Data Mining*. Foram também expostas as principais frentes de investigação neste tópico, bem como métricas e metodologias de avaliação para os métodos abordados, terminando o *survey* com um estudo das tendências da literatura na área nos últimos dez anos e enumeração das principais questões em aberto e *future work*.

A análise da literatura apontou haver um crescente interesse na área (dado que o número de publicações foi crescendo na maior parte das vezes ao longo dos anos), bem como um grande número de questões possíveis para trabalho futuro, o que caracteriza a área como bastante atrativa para novos investigadores.

Referências

1. Joao Gama. *Knowledge discovery from data streams*. CRC Press, 2010.
2. *Twitter API*. `https://dev.twitter.com/streaming/overview/connecting`. Accessed: 2016-05-22.
3. Brian Babcock et al. «Models and issues in data stream systems». Em: *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM. 2002, pp. 1–16.
4. Mohamed Medhat Gaber, Arkady Zaslavsky e Shonali Krishnaswamy. «Mining data streams: a review». Em: *ACM Sigmod Record* 34.2 (2005), pp. 18–26.
5. Shifei Ding et al. «Research on data stream clustering algorithms». Em: *Artificial Intelligence Review* 43.4 (2015), pp. 593–600.
6. Daniel Barbará. «Requirements for clustering data streams». Em: *ACM sIGKDD Explorations Newsletter* 3.2 (2002), pp. 23–27.
7. Vincent Lemaire, Christophe Salperwyck e Alexis Bondu. «A survey on supervised classification on data streams». Em: *Business Intelligence*. Springer, 2015, pp. 88–125.
8. Moses Charikar, Kevin Chen e Martin Farach-Colton. «Finding frequent items in data streams». Em: *Automata, languages and programming*. Springer, 2002, pp. 693–703.
9. Shriguru Nayak et al. «Exploring discriminative features for anomaly detection in public spaces». Em: *SPIE Defense+ Security*. International Society for Optics e Photonics. 2015, pp. 946403–946403.
10. Radhika Kotecha e Sanjay Garg. «Data streams and privacy: Two emerging issues in data classification». Em: *2015 5th Nirma University International Conference on Engineering (NUiCONE)*. IEEE. 2015, pp. 1–6.
11. Sakthithasan Sripirakas e Russel Pears. «Mining recurrent concepts in data streams using the discrete fourier transform». Em: *Data warehousing and knowledge discovery*. Springer, 2014, pp. 439–451.
12. Abad Miguel Ángel, Gomes João Bártolo e Menasalvas Ernestina. «Predicting recurring concepts on data-streams by means of a meta-model and a fuzzy similarity function». Em: *Expert Systems with Applications* 46 (2016), pp. 87–105.
13. FJ Villanueva et al. «Data stream visualization framework for smart cities». Em: *Soft Computing* (2015), pp. 1–11.
14. Yan Liu et al. «A dynamic assignment scheduling algorithm for big data stream processing in mobile internet services». Em: *Personal and Ubiquitous Computing* (2016), pp. 1–11.
15. Chetna Kaushal e Harpreet Singh. «Comparative study of recent sequential pattern mining algorithms on web clickstream data». Em: *2015 IEEE Power, Communication and Information Technology Conference (PCITC)*. IEEE. 2015, pp. 652–656.
16. Tiziano De Matteis e Gabriele Mencagli. «Keep calm and react with foresight: strategies for low-latency and energy-efficient elastic data stream processing». Em: *Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. ACM. 2016, p. 13.

17. Shree Garg, Sateesh K Peddoju e Anil K Sarje. «Scalable P2P bot detection system based on network data stream». Em: *Peer-to-Peer Networking and Applications* (2016), pp. 1–17.
18. Simon Fong et al. «Improvised methods for tackling big data stream mining challenges: case study of human activity recognition». Em: *The Journal of Supercomputing* (2016), pp. 1–33.
19. Matthias Moi et al. «Strategy for Processing and Analyzing Social Media Data Streams in Emergencies». Em: ().
20. Karthik Subbian, Charu C Aggarwal e Jaideep Srivastava. «Querying and Tracking Influencers in Social Streams». Em: *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM. 2016, pp. 493–502.
21. Ei Khaing Win et al. «A stream merge method to reduce load for sensor data stream delivery». Em: *2015 IEEE 4th Global Conference on Consumer Electronics (GCCE)*. IEEE. 2015, pp. 120–121.
22. Sonali Agarwal e Bakshi Rohit Prasad. «High speed streaming data analysis of web generated log streams». Em: *2015 IEEE 10th International Conference on Industrial and Information Systems (ICIIS)*. IEEE. 2015, pp. 413–418.
23. Max Zimmermann, Eirini Ntoutsi e Myra Spiliopoulou. «Extracting opinionated (sub) features from a stream of product reviews using accumulated novelty and internal re-organization». Em: *Information Sciences* 329 (2016), pp. 876–899.
24. Vladimir Nikulin, Tian-Hsiang Huang e Jian-De Lu. «Mining Shoppers Data Streams to Predict Customers Loyalty». Em: *Intelligent Systems and Knowledge Engineering (ISKE), 2015 10th International Conference on*. IEEE. 2015, pp. 26–33.
25. Mehdi Boukhechba et al. «Online prediction of people's next point-of-interest: concept drift support». Em: *Human Behavior Understanding*. Springer, 2015, pp. 97–116.