A sketch for the KS test for Big Data

Thalis D. Galeno[1], João Gama[2], and Douglas O. Cardoso[1]

[1] Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, Brasil
thalis.galeno@aluno.cefet-rj.br, douglas.cardoso@cefet-rj.br
[2] Universidade do Porto, Portugal
jgama@fep.up.pt

Abstract. Motivated by the challenges of Big Data, this paper presents an approximative algorithm to assess the Kolmogorov-Smirnov test. This goodness of fit statistical test is extensively used because it is non-parametric. This work focuses on the one-sample test, which considers the hypothesis that a given univariate sample follows some reference distribution. The method allows to evaluate the departure from such a distribution of a input stream, being space and time efficient. We show the accuracy of our algorithm by making several experiments in different scenarios: varying reference distribution and its parameters, sample size, and available memory. The performance of rival methods, some of which are considered the state-of-the-art, were compared. It is demonstrated that our algorithm is superior in most of the cases, considering the absolute error of the test statistic.

CCS Concepts: • **Information systems** → **Data stream mining**; • **Mathematics of computing** → **Nonparametric statistics**.

Keywords: data streams, incremental learning, kolmogorov-smirnov

1. INTRODUÇÃO

O aumento exponencial dos dispositivos computacionais e eletrônicos tornou o *Big Data* uma realidade, onde um de seus maiores desafios é o armazenamento de informações. A coleta de dados na Internet se tornou uma tarefa praticamente impossível de ser feita sem algum tipo de tratamento para sumarização dos dados. Mesmo tarefas primárias como a detecção de mudanças na distribuição dos dados, em amostras incrementalmente estabelecidas, se torna um grande desafio por conta do descarte de informações relevantes [Hu et al. 2020; Kifer et al. 2004]. Por exemplo, ao analisar o intervalo de chegada de pacotes em uma rede de alta velocidade [Claffy et al. 1993], ou o intervalo entre sinais de um certo tipo emitidos quando um meteoro passa por um telescópio [Barbosa et al. 2016], queremos identificar períodos em que a distribuição desse intervalo é igual, sendo assim preciso armazenar quantidades gigantescas de informação. Em muitas aplicações é impossível guardar tanta informação, pois o volume de dados é algumas ordens de magnitude maior que a memória disponível. Para lidar com tal quantidade de dados são necessários algoritmos eficientes, capazes de operar em tempo real e com um uso razoável dos recursos [Bifet 2017]. Para solucionar o problema é necessário trabalhar sobre a distribuição empírica dos dados. Este artigo apresenta um método para detectar mudanças ante a uma distribuição presumida de uma amostra conhecida, através do teste de *Kolmogorov-Smirnov* (KS).

O teste KS é um método estatístico para avaliar se uma amostra segue uma determinada distribuição. Esse teste é muito útil por ser não-paramétrico, ou seja, pode ser aplicado assumindo que os dados obedeçam qualquer tipo de distribuição. O teste é calculado basicamente avaliando a densidade de subamostras incrementais, o que se torna um problema quando se trata de amostras grandes. Logo,

Thalis D. Galeno participou desta pesquisa na condição de bolsista de iniciação científica da FAPERJ (processo E-26/200.693/2020).

uma característica idealizada é que se tenha um algoritmo cujo processamento tem custo constante para cada observação, fazendo com que o problema seja ainda mais complexo. Felizmente uma solução nestes moldes já existe [Gonzalez et al. 1978], apesar de não ser incremental. Também existem trabalhos na literatura que viabilizam o cálculo do teste de forma incremental mas aproximada de diversas formas [dos Reis et al. 2016; Nguyen 2018]. Todavia melhorar a estimativa dos resultados exatos do teste KS em tais condições ainda é um grande desafio computacional.

Neste trabalho propusemos um algoritmo incremental para cálculo do teste KS uniamostral, isto é, considerando as observações dadas e uma distribuição de referência. Para uma amostra com n itens, o método é capaz, dependendo da escolha de uma estrutura de dados auxiliar, de processar cada item em tempo $O(\log m)$ e obter o valor da estatística avaliada no teste KS em $O(1)$. O parâmetro m controla a memória utilizada de forma independente do tamanho da amostra, podendo ser consideravelmente menor que n. Ainda conforme tal estrutura de dados, pode se ter uma complexidade de memória de $O(m)$. Nos testes realizados em diversos cenários, o algoritmo apresentou, na maior parte destes, resultados superiores a outras abordagens para o mesmo fim, ainda que tomando por base o atual estado-da-arte.

O restante do artigo é organizado da seguinte forma. Na seção 2 apresentamos uma visão geral dos conceitos necessários para o entendimento do método proposto. Alguns trabalhos diretamente relacionados com o tema deste artigo são descritos na seção 3. O algoritmo proposto e seu funcionamento é descrito na seção 4. Testes e comparações feitas com outros métodos são apresentados na seção 5. Por fim, apresentamos as considerações finais e trabalhos futuros na seção 6.

2. PRELIMINARES

Visando um melhor entendimento do método proposto, esta seção tem o objetivo de apresentar os conceitos básicos necessários. O conceito de população pode ser definido como o conjunto de itens que interessa para um experimento. Um subconjunto dessa população é conhecido como amostra. O espaço amostral Ω é o conjunto de resultados possíveis para um experimento. Uma variável aleatória X é uma função que mapeia o resultado de um experimento em um único valor real, isto é $X : \Omega \to \mathbb{R}$. Uma função de distribuição, ou função de distribuição acumulada (*Cumulative Distribution Function* - CDF), é uma função que define a probabilidade de uma observação ser menor ou igual a um certo limite, definida como $F(x) = \Pr[X \leq x]$.

Quantis são usados para resumir ou caracterizar amostras de forma mais robusta a *outliers* que as opções clássicas de média e variância. Eles são uma ferramenta fundamental para análises estatísticas. O quantil $Q(\phi)$ pode ser interpretado como a função inversa da função de distribuição $F(x)$ onde $Q(F(x)) = x$. Outro conceito importante é o conhecido como função de distribuição empírica (*Empirical Cumulative Distribution Function* - ECDF) onde o seu valor em um certo ponto x, dada uma amostra A, é definido como

$$F_A(x) = \frac{|\{a \in A : a \leq x\}|}{|A|} \ .$$

Avaliar se há uma relação generativa entre uma amostra com uma distribuição de referência utilizando a ECDF da primeira e a CDF da última é a ideia central do teste de *Kolmogorov-Smirnov* (KS), na sua versão uniamostral. O teste KS é baseado na estatística D, também chamada de distância KS, que é capaz de representar o grau de associação entre uma distribuição de referência e uma amostra qualquer. Seu valor é a maior distância entre a CDF e a ECDF: $D = \sup_x |F_A(x) - F(x)|$. Para uma amostra $A = (X_1, X_2, ..., X_n)$ em ordem não-decrescente (ou seja, se $i \geq j$, então $X_i \geq X_j$) e uma CDF $F(x)$, a distância KS é determinada pela fórmula

$$D = \max_i \left(\frac{i}{n} - F(X_i), F(X_i) - \frac{i-1}{n} \right) \ ,$$

sendo $i/n = F_A(X_i)$ e $(i-1)/n = F_A(X_i - \epsilon)$ os valores na vizinhança de cada descontinuidade da ECDF, que é composta por vários "degraus". O algoritmo 1 descreve o pseudocódigo para o cálculo da distância KS de uma forma trivial, para uma amostra A qualquer, sem a restrição de que esta esteja ordenada *a priori*. Ele tem uma complexidade de memória e de tempo, respectivamente, de $O(n)$ e $O(n \log n)$, sendo dominado pela operação de ordenação da amostra usada como entrada.

Algoritmo 1 Cálculo da distância KS

1: Seja $A = (X_1, X_2, ..., X_n)$ a amostra a ser avaliada
2: $D = -\infty$
3: ordena(A)
4: **Para todo** i de 1 até n **:**
5: $\quad D = \max\{D, i/n - F(X_i), F(X_i) - (i-1)/n\}$

3. TRABALHOS RELACIONADOS

Um algoritmo para cálculo do teste KS em tempo linear foi apresentado por Gonzalez et al. [1977], utilizando três vetores auxiliares e uma estrutura de *buckets* para armazenar informações da amostra e assim obter o valor exato da estatística D. Seu desempenho se deve ao fato do método não precisar ordenar as amostras, fazendo o processamento passando por todos os itens duas vezes. Em contrapartida utiliza uma quantidade de memória maior em relação a forma trivial, ainda que por um fator multiplicativo constante. Posteriormente foi apresentado por Gonzalez et al. [1978] o APPROX_KS, um algoritmo muito similar para calcular a distância KS de forma aproximativa, também com tempo linear porém com uma complexidade de memória de $O(m)$, sendo m um parâmetro que define o número de buckets utilizados assim como a precisão da aproximação feita.

Mais recentemente Lall [2015] apresentou um algoritmo para a realização do teste KS voltado para dados em fluxo contínuo (*streams*), ou seja, cujo processamento deve ser feito de forma incremental e cujas complexidades de tempo e de memória idealmente sejam constantes ante ao número de observações processadas. O algoritmo é baseado no uso de qualquer estrutura de dados para estimação (*sketch*) de quantis amostrais, o que permite também aproximar a "inversa" ECDF e assim também a distância KS, tendo uma complexidade de tempo e espaço de $O(\sqrt{n} \log n)$ para uma estimativa considerada de boa confiabilidade, conforme sugerido pelo autor do método. Outra abordagem, proposta por dos Reis et al. [2016], é baseada em uma *treap*, que é uma árvore binária de busca randomizada. Esta realiza as operações de atualização em tempo $O(\log n)$ no caso médio, e consulta do valor da distância KS em $O(1)$. Este método se assemelha àquele aqui proposto por seu custo computacional. Entretanto sua forma de estimar o valor de D é um tanto menos direta e com custos computacionais mais altos no pior caso. Outra forma de realizar o teste KS uniamostral para *streams* foi proposta por Nguyen [2017], sendo este trabalho o primeiro a relatar testes com amostras de tamanho 10^9, visando prover uma noção mais realística dos resultados no contexto de Big Data. No ano seguinte Nguyen [2018] realizou modificações no método tendo evoluído para realizar o teste KS biamostral. Raab et al. [2020] foi o primeiro a apresentar um algoritmo utilizando o teste KS para detecção de mudanças recorrentes em *streams*.

O cálculo de quantis em estatística é muito utilizado, não só em computação, mas nas mais diversas áreas de conhecimento. No contexto do teste KS, os quantis são especialmente úteis, pois representam a inversa de uma CDF, ou ECDF no caso de quantis amostrais, podendo ser utilizados para se obter o valor da distância KS assim como feito por Lall [2015]. Greenwald and Khanna [2001] apresentaram um algoritmo online para estimação de quantis, conhecido como algoritmo GK, capaz de lidar com Big Data fazendo um uso de memória de $O(\frac{1}{\epsilon} \log(\epsilon n))$, para responder a consultas com uma precisão de ϵn. O algoritmo apresentado não precisa saber a priori o tamanho da amostra, sendo essa uma característica desejável para aplicação em *streams*. Outro importante trabalho é de Shrivastava

et al. [2004] onde foi introduzido o Quantile Digest ou Q-Digest, o qual originalmente é voltado para quantis em um espaço amostral discreto de tamanho U. O Q-Digest utiliza memória de $O(\frac{1}{\epsilon}\log U)$ para sua estrutura de dados: uma árvore binária com nós que representam buckets de tamanhos variáveis, cujo tamanho é mantido aproximadamente igual por meio de operações de *merge* e *split* destes. Conforme notado por Buragohain and Suri [2009] o método GK lida melhor com entradas mais variadas e esparsas, enquanto o Q-Digest é superior se tratando do cenário oposto, i.e. um espaço amostral de entrada limitado e *streams* densas. Mais recentemente Masson et al. [2019] apresentou mais um algoritmo para quantis em *streams*, o DDSketch, o qual pode ser considerado como o estado-da-arte entre estruturas de dados para esta finalidade.

4. METODOLOGIA

Nesta seção descrevemos o algoritmo proposto para cálculo do valor da estatística D do teste KS para dados unidimensionais, chamado de GreedyKS, apresentando seu pseudocódigo, complexidade e suas particularidades. O GreedyKS é inspirado nos algoritmos de Gonzalez et al. [1977; 1978], sendo incremental e aproximativo. Assim como seus antecessores, atribui cada observação de entrada a um dos elementos de uma sequência de buckets, mantendo para um certo bucket de índice b as seguintes informações: a contagem de elementos (num_b), o maior ($most_b$) e o menor ($least_b$) valor de CDF. O número de buckets utilizados (m) é passado como parâmetro na inicialização e pode ser significativamente menor que o número de observações eventualmente processadas.

A própria função de distribuição cumulativa de referência é utilizada para determinar o bucket ao qual cada observação é associada. O método mantém duas variáveis DP e DN, para registrar as maiores diferenças da ECDF para a CDF e vice-versa, respectivamente. A cada novo elemento processado o algoritmo atualiza os três atributos do bucket correspondente, calcula as diferenças entre ECDF e CDF na região respectiva ao bucket onde o novo elemento foi inserido e, caso necessário, atualiza as variáveis DP e DN. Importante notar que ao atualizar o valor dessas variáveis levando em conta somente o bucket modificado por último, podemos obter uma acurácia relativamente baixa. Para combater este problema também são armazenados os índices dos buckets que maximizaram as variáveis DP e DN para que estes sejam reanalisados em futuras iterações.

Ao fazer uma iteração do algoritmo, o novo elemento pode pertencer a qualquer um dos buckets. Para calcular as diferenças entre ECDF e CDF relativas a um bucket é necessário saber quantos elementos estão posicionados em buckets anteriores. Visando otimizar tal consulta, foi utilizada uma estrutura de dados para soma de prefixos [Pibiri and Venturini 2021] a fim de obter o número de elementos posicionados até um determinado bucket: por exemplo, árvore de Fenwick (também conhecida como *Binary Indexed Tree*, BIT) [Fenwick 1994], árvore de segmentos [Laaksonen 2017], ou decomposição raiz [Halim 2020]. Neste trabalho foi utilizada a estrutura BIT por conta das suas operações de atualização e consulta cujo custo computacional é de ordem logarítmica.

O algoritmo 2 descreve o pseudocódigo de uma iteração do GreedyKS. O valor da estatística D pode ser obtido em tempo constante fazendo a operação $\max\{DN, DP\}$. O custo computacional do algoritmo é predominantemente definido pela atualização e consulta de somas de prefixos, haja vista que as demais operações são atômicas, tendo custo de tempo constante. Consequentemente cada iteração tem uma complexidade de tempo de $O(\log m)$.

5. AVALIAÇÃO EXPERIMENTAL

O algoritmo proposto foi avaliado experimentalmente utilizando dados sintéticos e foi comparado com outros métodos já conhecidos para testar sua acurácia. Foram medidos os erros absolutos (o módulo da diferença entre os valores exatos e as aproximações obtidas pelos métodos) da estatística D do teste KS uniamostral. O código foi todo escrito na linguagem Python. Todos os experimentos foram executados em um Intel(R) Xeon(R) E3-1240 v6 3.70GHz com 64GB de RAM rodando o sistema operacional

Algoritmo 2 Iteração do GreedyKS

Seja x o elemento a ser processado a seguir
Seja n a quantidade de elementos processados até então
Seja F a CDF de referência
$f = F(x)$
$n = n + 1$
$b = \lfloor f \cdot m \rfloor$
$num_b = num_b + 1$ ▷ Atualização de soma de prefixo
$most_b = \max\{most_b, f\}$
$least_b = \min\{least_b, f\}$
$DN = least_{oldN} - (\sum_{i=0}^{oldN-1} num_i)/n$ ▷ Consulta de soma de prefixo
$DN' = least_b - (\sum_{i=0}^{b-1} num_i)/n$ ▷ Consulta de soma de prefixo
if $DN' > DN$ **then**
    $DN = DN'$
    $oldN = b$
$DP = (\sum_{i=0}^{oldP} num_i)/n - most_{oldP}$ ▷ Consulta de soma de prefixo
$DP' = (\sum_{i=0}^{b} num_i)/n - most_b$ ▷ Consulta de soma de prefixo
if $DP' > DP$ **then**
    $DP = DP'$
    $oldP = b$

Ubuntu 18.04. Foram utilizadas amostras de distribuições Normal ($\mathcal{N}(\mu, \sigma^2)$ representa a distribuição Normal com média μ e variância σ^2), Uniforme ($\mathcal{U}_{[a,b]}$ é a distribuição Uniforme no intervalo $[a, b]$) e Pareto ($Par(\alpha, \beta)$ é a distribuição de Pareto com escala β e formato α), reproduzindo o protocolo experimental de Lall [2015]. Para todos os testes feitos, foram geradas 100 amostras independentes e relatadas as médias dos resultados. Com exceção dos casos indicados de outra forma, cada amostra possuía 10^5 elementos e o espaço em memória disponibilizado para realizar o teste KS era da ordem de 1% do tamanho da amostra. Em outras palavras, para amostras de tamanho 10^5, utilizar memória de 1% significa configurar m como 10^3. Os experimentos foram realizados utilizando distribuições muito próximas propositalmente, com o intuito de explorar a capacidade de detecção dos métodos na menor variação de distribuição. O caso onde as amostras possuem funções de distribuição muito distantes é relativamente fácil de detectar mesmo com substanciais restrições de memória.

Além do GreedyKS, os seguintes métodos foram testados:

— Lall + DDSketch [Lall 2015; Masson et al. 2019] - foi incluído por ser considerado uma referência do estado-da-arte.
— APPROX_KS [Gonzalez et al. 1978] - apesar de não ser voltado para uso em *streams*, utilizamos esse método apenas para servir como referencial enquanto predecessor do GreedyKS.
— *(Reservoir) Sampling* [Bifet 2017] - de forma iterativa é feita uma amostragem da entrada e calculado o valor exato da estatística D da subamostra. Representa a solução mais básica possível.
— GreedyKS* - versão do GreedyKS com número de buckets igual ao tamanho da amostra. Ou seja, sem restrição quanto ao uso de memória. O objetivo de incluir essa alternativa é medir a influência da restrição de memória sobre o resultado obtido.

A figura 1 ilustra uma comparação voltada para a restrição da memória disponível. No primeiro cenário, é avaliada a distância KS entre uma distribuição $\mathcal{N}(0, 1)$ e amostras distribuídas conforme uma $\mathcal{N}(0.1, 1)$. Os outros dois cenários são semelhantes, todavia considerando outras distribuições. De modo geral os métodos apresentaram comportamentos semelhantes para os três tipos de distribuição, com exceção do Lall + DDSketch que apresentou um desempenho suavemente melhor na distribuição

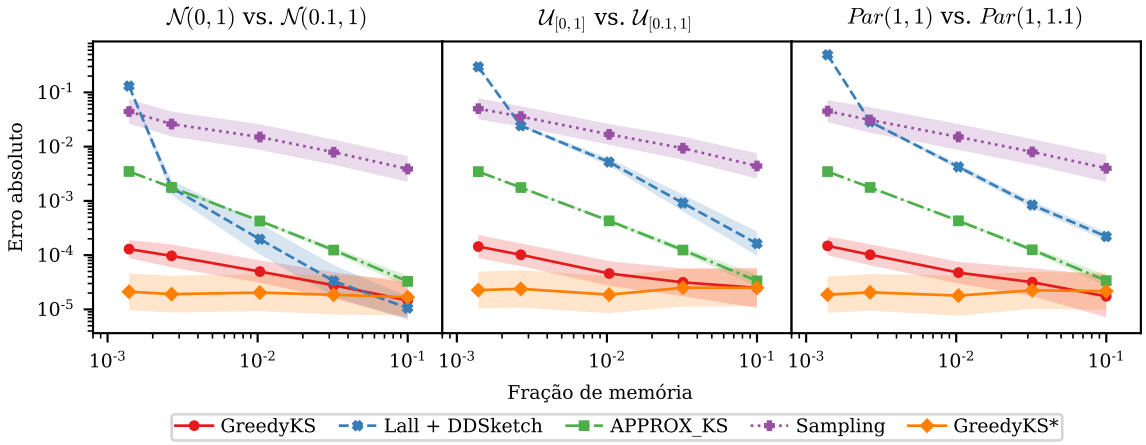

Fig. 1: Variando percentuais de memória com 10^5 elementos e diferentes distribuições.

Normal em relação as outras distribuições. Os métodos GreedyKS* e GreedyKS apresentaram um erro baixo e estável mesmo para uma utilização da ordem de 0.1% de memória. Por outro lado o Lall + DDSketch mostra um desempenho semelhante aos outros métodos com 10% de uso de memória e distribuição Normal, porém sua acurácia é prejudicada a medida que as restrições de memória aumentam.

Na figura 2 estão ilustrados os resultados quando foram realizadas variações no tamanho da amostra, para observar como os métodos se comportam com o aumento do volume de dados. O comportamento de queda dos algoritmos com o aumento do conjunto se deve ao fato de que com mais itens no volume total, mais itens são usados para calcular o valor da estatística do teste KS, obtendo então uma melhor acurácia. Mais uma vez os métodos não tiveram grandes modificações na comparação entre distribuições. O método de amostragem se manteve com o pior desempenho e GreedyKS* como o melhor em quase todos os tamanhos testados. Logo abaixo, com um desempenho melhor que na amostragem, o APPROX_KS se manteve estável com pouca variação. Em seguida, o Lall + DDSketch se mostrou o melhor dentre os métodos respeitando a restrição de memória na ordem de 1% para amostras de tamanho 10^3, mas o GreedyKS logo assume esse posto conforme o volume de dados aumenta.

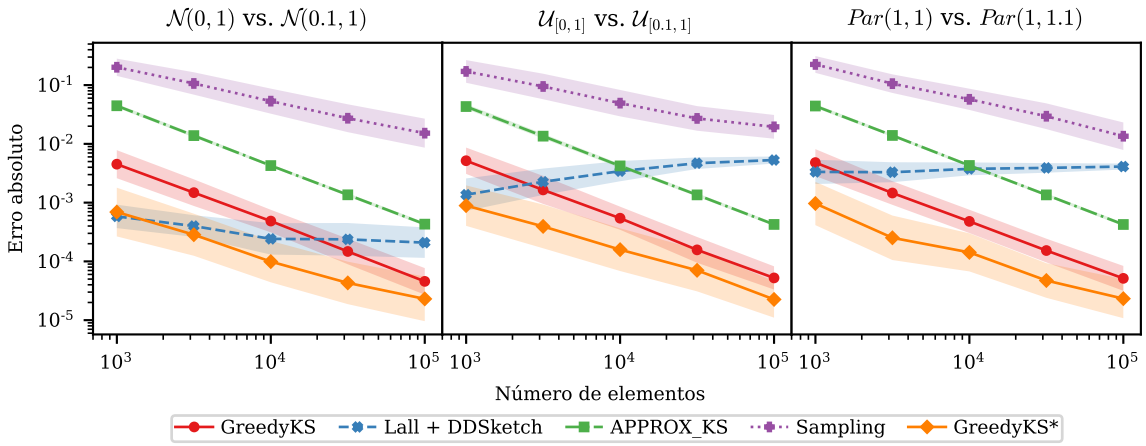

Fig. 2: Variando tamanho dos conjuntos de dados com memória na ordem de 1% e diferentes distribuições.

Em seguida testamos como os métodos se comportam com diferenças na distribuição, conforme ilustrado na figura 3. Os métodos se mostraram bem estáveis em todas as distribuições, obtendo um erro praticamente igual em toda faixa de valores experimentados. Todos obtiveram praticamente os mesmos valores de erro nas três distribuições, com exceção do Lall + DDSketch que teve melhor desempenho na distribuição Normal. Novamente, os métodos Sampling e GreedyKS* foram os de pior e melhor desempenho respectivamente. O APPROX_KS teve um desempenho superior ao Sampling, obtendo um resultado praticamente constante. O Lall + DDSketch teve um resultado que até chegou perto do GreedyKS na distribuição Normal com média 0.1, mas o GreedyKS logo aumenta a vantagem, principalmente nas distribuições Uniforme e Pareto. O GreedyKS e GreedyKS* tiveram um resultado muito similar nas diversas situações consideradas, indicando que o GreedyKS sofre pouca influência da restrição de memória: ambos mantiveram o erro absoluto abaixo de 10^{-4}.

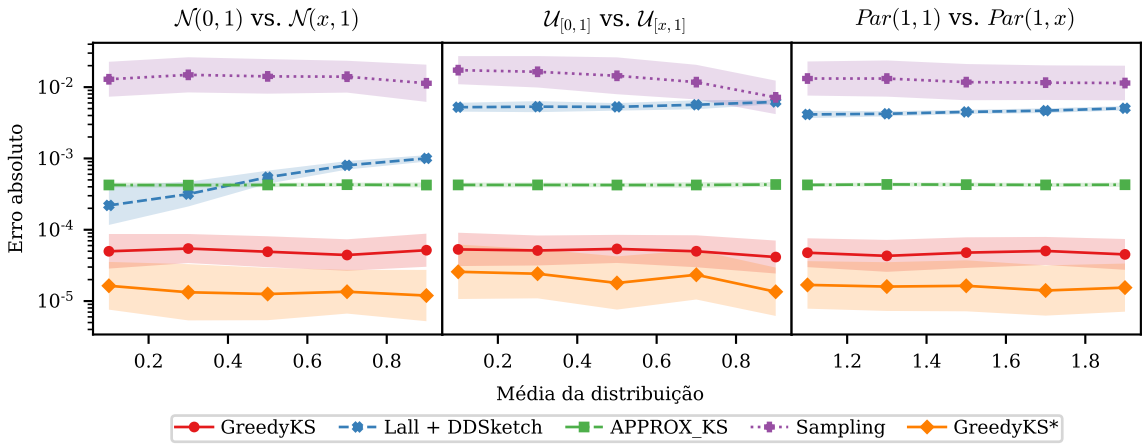

Fig. 3: Variando média da distribuição com 10^5 elementos e aproximadamente 1% de memória em diferentes distribuições.

6. CONCLUSÃO

Neste artigo, consideramos o problema de calcular computacionalmente o teste de Kolmogorov-Smirnov de forma incremental em Big Data. Nós apresentamos um algoritmo incremental para cálculo da versão uniamostral do teste. O método utiliza o cálculo convencional do teste em conjunto com técnicas de computação e uma estrutura de dados auxiliar, permitindo seu uso com altíssimas restrições de memória. Além disso, mostramos que o algoritmo tem um desempenho notadamente superior tanto a técnicas simples, como a amostragem, quanto a outros métodos mais refinados. Isto foi confirmado por meio de experimentos com diferentes distribuições, tamanhos de amostra e restrições de memória. Segundo tais testes, o GreedyKS obteve maior vantagem em relação a rivais para as maiores amostras, como as de 10^5 observações, e altas restrições de uso de memória, como da ordem de 0.1% da entrada.

Há vários pontos passíveis de melhorias que ainda persistem. Um deles é realizar testes com amostras de tamanhos maiores e avaliar o desempenho usando dados reais. No futuro planejamos encontrar uma garantia matemática de performance para o algoritmo e adicionar outras técnicas de computação para aumentar a acurácia. Em trabalhos futuros podemos evoluir o algoritmo para desconsiderar dados defasados e assim processar dados em fluxo contínuo. Outro caminho seria encontrar uma forma de alterar o algoritmo e chegar a uma versão para cálculo do teste biamostral.

REFERENCES

Barbosa, D., Barraca, J. P., Carvalho, B., Maia, D., Gupta, Y., Natarajan, S., Roux, G. L., and Swart, P. A cyber infrastructure for the SKA Telescope Manager. In *Software and Cyberinfrastructure for Astronomy IV*, G. Chiozzi and J. C. Guzman (Eds.). Vol. 9913. International Society for Optics and Photonics, SPIE, Edinburgh, United Kingdom, pp. 213 – 224, 2016.

Bifet, A. *Machine learning for data streams: with practical examples in MOA*. Adaptive computation and machine learning series. MIT Press, Cambridge, Massachusetts, 2017.

Buragohain, C. and Suri, S. Quantiles on Streams. In *Encyclopedia of Database Systems*, L. LIU and M. T. ÖZSU (Eds.). Springer US, Boston, MA, pp. 2235–2240, 2009.

Claffy, K. C., Polyzos, G. C., and Braun, H.-W. Application of sampling methodologies to network traffic characterization. *ACM SIGCOMM Computer Communication Review* 23 (4): 194–203, Oct., 1993.

dos Reis, D. M., Flach, P., Matwin, S., and Batista, G. Fast Unsupervised Online Drift Detection Using Incremental Kolmogorov-Smirnov Test. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. Association for Computing Machinery, New York, NY, USA, pp. 1545–1554, 2016.

Fenwick, P. M. A new data structure for cumulative frequency tables. *Software: Practice and Experience* 24 (3): 327–336, 1994.

Gonzalez, T., Sahni, S., and Franta, W. R. An Efficient Algorithm for the Kolmogorov-Smirnov and Lilliefors Tests. *ACM Transactions on Mathematical Software* 3 (1): 60–64, Mar., 1977.

Gonzalez, T., Sahni, S., and Franta, W. R. An efficient approximate algorithm for the Kolmogorov—Smirnov and Lilliefors tests. *Journal of Statistical Computation and Simulation* 6 (3-4): 257–263, Jan., 1978.

Greenwald, M. and Khanna, S. Space-efficient online computation of quantile summaries. *ACM SIGMOD Record* 30 (2): 58–66, May, 2001.

Halim, S. *Competitive programming 4 : the lower bound of programming contests in the 2020s*. Lulu.com, Singapore, 2020.

Hu, H., Kantardzic, M., and Sethi, T. S. No free lunch theorem for concept drift detection in streaming data classification: A review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 10 (2): e1327, 2020.

Kifer, D., Ben-David, S., and Gehrke, J. Detecting change in data streams. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*. VLDB '04. VLDB Endowment, Toronto, Canada, pp. 180–191, 2004.

Laaksonen, A. *Guide to Competitive Programming - Learning and Improving Algorithms Through Contests*. Undergraduate Topics in Computer Science. Springer, Cham, Switzerland, 2017.

Lall, A. Data streaming algorithms for the kolmogorov-smirnov test. In *Proceedings of the 2015 IEEE International Conference on Big Data (Big Data)*. BIG DATA '15. IEEE Computer Society, USA, pp. 95–104, 2015.

Masson, C., Rim, J. E., and Lee, H. K. DDSketch: A fast and fully-mergeable quantile sketch with relative-error guarantees. *Proceedings of the VLDB Endowment* 12 (12): 2195–2205, Aug., 2019.

Nguyen, H. D. A stream-suitable kolmogorov-smirnov-type test for big data analysis, 2017.

Nguyen, H. D. A Two-Sample Kolmogorov-Smirnov-Like Test for Big Data. In *Data Mining*, Y. L. Boo, D. Stirling, L. Chi, L. Liu, K.-L. Ong, and G. Williams (Eds.). Communications in Computer and Information Science. Springer, Singapore, pp. 89–106, 2018.

Pibiri, G. E. and Venturini, R. Practical trade-offs for the prefix-sum problem. *Software: Practice and Experience* 51 (5): 921–949, 2021.

Raab, C., Heusinger, M., and Schleif, F.-M. Reactive Soft Prototype Computing for Concept Drift Streams. *Neurocomputing* vol. 416, pp. 340–351, Nov., 2020.

Shrivastava, N., Buragohain, C., Agrawal, D., and Suri, S. Medians and beyond: New aggregation techniques for sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*. SenSys '04. Association for Computing Machinery, New York, NY, USA, pp. 239–249, 2004.