Olá mundo.

Circuitos Digitais - Introdução

Paulo Ricardo Lisboa de Almeida

DSBD

UFPR

Professor

Paulo Ricardo Lisboa de Almeida

Doutor em Ciência da Computação, Bel. em Eng. da Computação.

paulorla@ufpr.br

prlalmeida.com.br

www.linkedin.com/in/paulorla

Professor - Pesquisa

Aprendizado de máquina.

Machine Learning para fluxos de dados.

Cidades inteligentes.

Paulo Grégio Zanata



DSBD + Secret + Hipes

Nosso Hardware

4 servidores:

- 328 processadores.
- 3,3 TB de DRAM.
- GPUs para criação de modelos de IA
  - 4 GPUs NVidia A5000.
  - 1 GPU NVidia A6000
  - 43.520 CUDA Cores.
  - 144 GB de memória de vídeo.

Efficient Prequential AUC-PR Computation

Artigo - IEEE ICMLA 2023.

Florida, Estados Unidos.

SAMSUNG

David Paulo Grégio Zanata




Efficient Prequential AUC-PR Computation

David L. Pereira Gomes, André Grégio, Marco A. Zanata Alves, Paulo R. Lisboa de Almeida
Department of Informatics – Federal University of Paraná (UFPR) – Curitiba, PR - Brazil
david.gomes@ufpr.br, gregio@ufpr.br, mazalves@ufpr.br, paulorla@ufpr.br

***Abstract*—When dealing with classification problems for data streams, we often need to compute the classification metrics in a prequential manner. The Area Under the Precision-Recall Curve (AUC-PR) metric is extensively used in imbalanced classification scenarios, where the negative class outnumbers the positive one. Despite its advantages, it may be computationally expensive to recompute that metric every time a new test instance becomes available. In this work, we present an efficient algorithm to compute the AUC-PR in a prequential way. Our proposed algorithm uses a self-balancing binary search tree to avoid the need to reorder the data when updating the AUC-PR value with the most recent data. Our experiments take into consideration six well-known, publicly available stream-based datasets. Our experiments show that our approach can be up to 13 times faster and use 12 times less energy than the traditional batch approach when considering a window of size 1,000.**

***Index Terms*—AUC-PR, prequential, stream, metrics**

I. Introduction

The massive amount of data produced by sensors, devices, and users poses a challenge to the application of classification algorithms whose output needs to be provided in real-time (e.g., critical systems, emergency diagnosis, security, threat detection, etc.). Those data are usually temporally-dependent, arriving at the classifier as a data stream.

Metrics for classification problems involving data streams, such as accuracy, F1-score, and Area Under the Precision-Recall Curve (AUC-PR), are often computed in a prequential manner, i.e., every time a new test instance becomes available. The reasoning behind the prequential calculation of those metrics is to allow for the monitoring of the classifier's performance over time, as well as quickly reacting to environmental changes (e.g., concept drifts), which may hinder the classification capability of a decision-support system.

Therefore, the prequential computation of classification metrics often requires computing them using a window W that contains the latest labeled data received. Thus, a metric must be recomputed on each update of this window, which may lead to an overhead that turns the classification of data streams into an overly expensive task, especially if we rely on computationally intensive metrics, such as the AUC-PR. The incurred overhead may increase costs (e.g., more computing power in servers) and the carbon footprint associated with this type of classification system.

In this work, we introduce an efficient algorithm to compute the AUC-PR for streams in a prequential manner (assuming a stream of instances, in which samples arrive for classification one at a time). The AUC-PR metric belongs to a family of metrics focused on imbalanced scenarios. To the best of our knowledge, this is the first algorithm to reduce the time complexity from $O(m \log m)$ to $O(m)$ when computing the AUC-PR metric for streams. In our experiments, the proposed algorithm was 13 times faster and used 12 times less energy when compared to the batch approach (i.e., recomputing the metric from scratch every time the window W is updated), often used when a prequential algorithm is unavailable. The main contributions of this paper are:

- An algorithm to calculate the AUC-PR for stream settings in a prequential way, focusing on its efficiency;
- The evaluation of our proposed algorithm and comparison with a widely used implementation of the metric that considers batch settings.

II. Background and Related Work

In this Section, we introduce concepts needed for properly understanding our proposed method, such as the prequential computation of metrics and the definition of the Precision-Recall Curve. We also present the related state-of-the-art work.

A. Batch versus Prequential Metrics

Data classification can be divided into two settings: batch (or static) learning or stream. In the former, we consider that the available data is limited to a "snapshot" of a certain period of time, whereas in the latter, we have to handle unlimited data continuously arriving at potentially high rates [1].

Under a static setting, we may create a classifier using a train set S_r, and test its performance using some metric in a test set S_t, where $S_r \cap S_t = \emptyset$. This approach is known as holdout or batch testing [2]. On the other hand, under a stream scenario, new instances arrive over time, making it impossible to have a fixed test set to assess the classifier's performance—especially under conditions where the problem may evolve.

In a stream scenario, it is common to define a window W containing the m latest instances received and compute the performance metrics using this window. Every time a new test instance arrives, this window is moved to accommodate the new instance, and the metrics are updated. This approach is known as the prequential computation of the metric [2].

For example, let's consider the stream at times t and $t+1$ in Figure 1, in which the metrics are computed within a window that contains the $m = 5$ latest instances. When a test instance $\mathbf{x}_{t+1}$ arrives at time $t+1$, the window is moved to accommodate this new instance, and the oldest instance in the window ($\mathbf{x}_{t-4}$) is removed from the window.

When the window moves, it is updated, and we may recompute the performance metrics using the entire window (i.e., the whole window is considered a batch). Besides being simple,

ICMLA 2023.

Professor - Pesquisa

Grupo DSBD.

dsbd.inf.ufpr.br

Tratamos mais sobre o assunto no final da disciplina.

Iniciação científica, TCC, Mestrado, Doutorado, ...

Caminho de Hardware

Circuitos Digitais *<- você está aqui*

Sistemas de numeração para inteiros
Portas lógicas
Circuitos
Memórias simples

Projeto

FPGAs e VHDL
Assembly básico
Processador Simples

Arquitetura de Computadores

Processador com Pipeline
Hierarquia de Memórias
Ponto Flutuante
Introdução ao paralelismo em hardware

Software Básico

Assembly (hard mode)
Representação de dados
Interface Hardware Software

Ementa

- Sistemas de numeração, conversão de bases.
- Aritmética binária: Soma, Subtração, Multiplicação.
- Equações booleanas, simplificação de álgebra booleana.
- Mapas de Karnaugh.
- Portas lógicas básicas.
- Blocos combinacionais: multiplexadores, demultiplexadores, decodificadores e seletores.
- Latches e flip-flops.
- Contadores síncronos e assíncronos.
- Máquinas de estado finito: projeto, codificação de estados, fatoração.

Moodle

A disciplina tem uma página oficial na UFPRVirtual.

Será o meio oficial de comunicação.

Atualize o seu e-mail.

Para se comunicar, você pode ir até a sala do professor, ou enviar e-mail.

paulorla@ufpr.br

Avaliação

1 prova: 35%

2 trabalhos: 30%

1 exercícios Moodle 5%

Mínimo de presenças: 75% -› **reprovado** automaticamente se não cumprir.

Aprovado se média ≥ 70.

Exame se nota ≥ 40. Nesse caso nota = (média + exame)/2. **Aprovado** se nota ≥ 5.

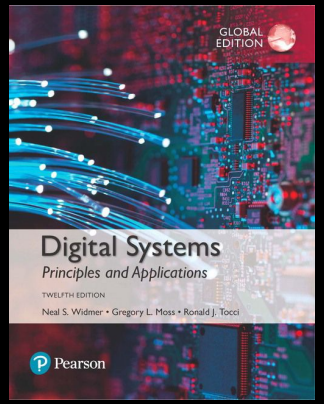
Entregas e Provas

Não serão aceitas entregas em atraso (exceto casos amparados pela UFPR).

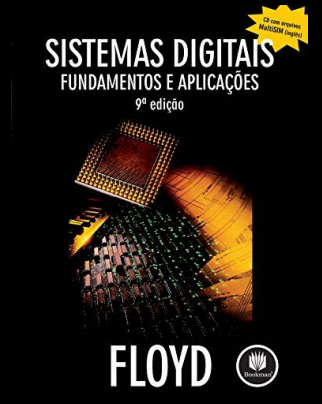
Em caso de plágio, **todos** envolvidos ficam com zero.

Bibliografia Básica

Ronald J. Tocci, Gregory L. Moss, Neal S. Widmer. Sistemas digitais. 10a ed. 2017.



Thomas Floyd. Widmer. Sistemas Digitais: Fundamentos e Aplicações. 2009.



Veja a bibliografia completa no plano da disciplina

Biblioteca Virtual

A UFPR dá acesso gratuito a uma biblioteca virtual a todos seus alunos.

Acesse via https://minhabiblioteca.ufpr.br/biblioteca

Dica: você vai precisar do seu e-mail @ufpr para acessar.

Ficha 2

Para mais detalhes, como bibliografia complementar e programa completo da disciplina, veja a Ficha 2.

É um plano, portanto pode sofrer alterações.

Em especial, as datas podem mudar.

Tudo será avisado com antecedência.

Bibliografia - Internet

Cuidado!

A internet é uma fonte importante de informações.

Bibliografia - Internet

Cuidado!

A internet é uma fonte importante de informações.

E uma fonte inesgotável de bobagens e pseudoespecialistas!

Seja criterioso ao pesquisar algum conceito na internet.

Na dúvida entre em contato com o professor.

Importância

Se você não souber o básico sobre como o computador funciona, você é um usuário de computadores, e não um Cientista da Computação / Informata Biomédico.

Importância

Se você não souber o básico sobre como o computador funciona, você é um usuário de computadores, e não um Cientista da Computação / Informata Biomédico.

Você pode precisar criar seu próprio hardware.

Importância

Se você não souber o básico sobre como o computador funciona, você é um usuário de computadores, e não um Cientista da Computação / Informata Biomédico.

Você pode precisar criar seu próprio hardware.

Saber como o hardware funciona afeta os seus programas de alto nível.

Importância

Se você não souber o básico sobre como o computador funciona, você é um usuário de computadores, e não um Cientista da Computação / Informata Biomédico.

Você pode precisar criar seu próprio hardware.

Saber como o hardware funciona afeta os seus programas de alto nível.

Processamento de alto desempenho exige conhecimentos detalhados sobre o funcionamento da máquina.

Importância

"... A professional in any field of computing should not regard the computer as just a black box that executes programs by magic. All students of computing should acquire some understanding and appreciation of a computer system's functional components, their characteristics, their performance, and their interactions ... in order to make best use of the software tools ..." (IEEE/ACM Computer Science Curriculum, 2008).

"...Um profissional de qualquer ramo da computação não deveria considerar o computador como uma simples caixa preta que executa programas por mágica. Todos estudantes de computação devem adquirir conhecimento e apreciação quanto aos componentes funcionais, características, performance e interações de sistemas computacionais ... a fim de fazer o melhor uso das ferramentas de software ..." (IEEE/ACM Computer Science Curriculum, 2008).

Importância

O mundo não é (nem de longe) feito somente de computadores desktop.

Que outros computadores você consegue enxergar no seu dia a dia?

Importância

Os conceitos da disciplina servem também como base para criação sistemas computacionais envolvendo:

Sistemas embarcados.

Sistemas baseados em microcontroladores.

Criação de circuitos integrados.

...

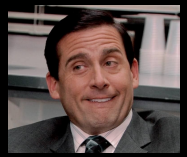
Um exemplo

```c
#include <stdio.h>

int main(){
   int valor = 2147483645;
   do{
       printf("Somando 1\n");
       valor = valor + 1;
       printf("valor atual: %d\n", valor);
   }while(getchar() != 's');

   return 0;
}
```

Um exemplo

Saída do Programa:

Somando 1
valor atual: 2147483646

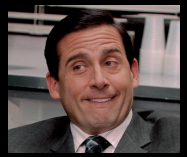


```c
#include <stdio.h>

int main(){
    int valor = 2147483645;
    do{
        printf("Somando 1\n");
        valor = valor + 1;
        printf("valor atual: %d\n", valor);
    }while(getchar() != 's');

    return 0;
}
```

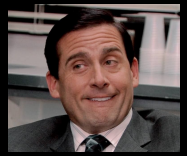
Um exemplo

Saída do Programa:

Somando 1
valor atual: 2147483646



Somando 1
valor atual: 2147483647

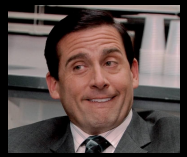


```c
#include <stdio.h>

int main(){
    int valor = 2147483645;
    do{
        printf("Somando 1\n");
        valor = valor + 1;
        printf("valor atual: %d\n", valor);
    }while(getchar() != 's');

    return 0;
}
```

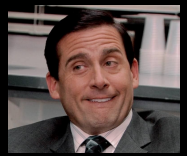
Um exemplo

Saída do Programa:

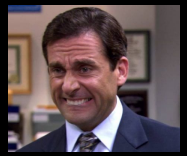
Somando 1
valor atual: 2147483646



Somando 1
valor atual: 2147483647



Somando 1
valor atual: -2147483648

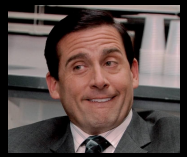


```c
#include <stdio.h>

int main(){
    int valor = 2147483645;
    do{
        printf("Somando 1\n");
        valor = valor + 1;
        printf("valor atual: %d\n", valor);
    }while(getchar() != 's');

    return 0;
}
```

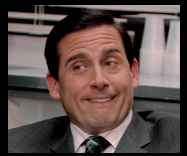
Um exemplo

Saída do Programa:

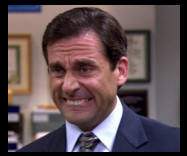
Somando 1
valor atual: 2147483646



Somando 1
valor atual: 2147483647



Somando 1
valor atual: **-2147483648**

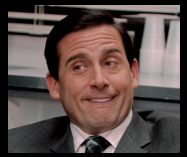


- Ao somar 1 o valor fica negativo!?

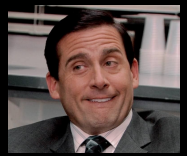
Um exemplo

Saída do Programa:

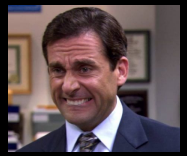
Somando 1
valor atual: 2147483646



Somando 1
valor atual: 2147483647



Somando 1
valor atual: **-2147483648**

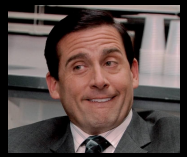


- Ao somar 1 o valor fica negativo!?
- Sempre vai ser esse valor?

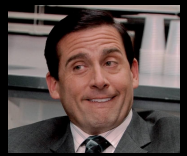
Um exemplo

Saída do Programa:

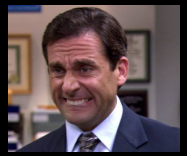
Somando 1
valor atual: 2147483646



Somando 1
valor atual: 2147483647



Somando 1
valor atual: **-2147483648**

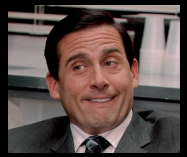


- Ao somar 1 o valor fica negativo!?
- Sempre vai ser esse valor?
- Por que ficou negativo?

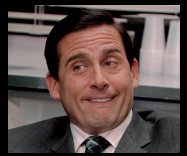
Um exemplo

Saída do Programa:

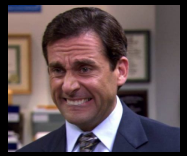
Somando 1
valor atual: 2147483646



Somando 1
valor atual: 2147483647



Somando 1
valor atual: **-2147483648**

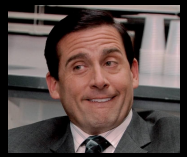


- Ao somar 1 o valor fica negativo!?
- Sempre vai ser esse valor?
- Por que ficou negativo?
- Será que nosso computador não sabe fazer contas simples?

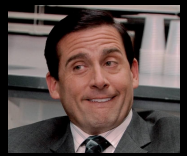
Um exemplo

Saída do Programa:

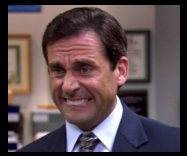
Somando 1
valor atual: 2147483646



Somando 1
valor atual: 2147483647



Somando 1
valor atual: **-2147483648**



- Ao somar 1 o valor fica negativo!?
- Sempre vai ser esse valor?
- Por que ficou negativo?
- Será que nosso computador não sabe fazer contas simples?

Esse e outros mistérios serão resolvidos no decorrer da disciplina.

Outro exemplo

```c
#include <stdio.h>

int main(){
   double teste = 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1;

   printf("%.16f\n", teste);

   return 0;
}
```

Outro exemplo

Aaaaargh!

Seu computador não sabe fazer contas simples!

Não é bem assim!

Vamos aprender o motivo durante a disciplina.

```c
#include <stdio.h>

int main(){
   double teste = 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1;

   printf("%.16f\n", teste);

   return 0;
}
```

Saída do Programa:

0.9999999999999999

Dica de leitura

Parker, M. Humble Pi: A Comedy of Maths Errors. Reino Unido: Penguin Books Limited. 2019.

É isso...

Perguntas?

Licença