Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Recomendação de Algoritmos em Fluxos de Dados com Mudança de Conceito

Jáder Martins Camboim de Sá

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Luís Paulo Faina Garcia

Brasília
2020

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Recomendação de Algoritmos em Fluxos de Dados com Mudança de Conceito

Jáder Martins Camboim de Sá

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Luís Paulo Faina Garcia (Orientador)
CIC/UnB

Prof. Dr. Donald Knuth
Stanford University

Dr. Leslie Lamport
Microsoft Research

Prof.a Dr.a Ada Lovelace
Coordenadora do Bacharelado em Ciência da Computação

Brasília, 05 de agosto de 2020

Dedicatória

Na *dedicatória* o autor presta homenagem a alguma pessoa (ou grupo de pessoas) que têm significado especial na vida pessoal ou profissional. Por exemplo (e citando o poeta): *Eu dedico essa música a primeira garota que tá sentada ali na fila. Brigado!*

Agradecimentos

Nos *agradecimentos*, o autor se dirige a pessoas ou instituições que contribuíram para elaboração do trabalho apresentado. Por exemplo: *Agradeço aos gigantes cujos ombros me permitiram enxergar mais longe. E a Google e Wikipédia.*

Resumo

O *resumo* é um texto inaugural para quem quer conhecer o trabalho, deve conter uma breve descrição de todo o trabalho (apenas um parágrafo). Portanto, só deve ser escrito após o texto estar pronto. Não é uma coletânea de frases recortadas do trabalho, mas uma apresentação concisa dos pontos relevantes, de modo que o leitor tenha uma ideia completa do que lhe espera. Uma sugestão é que seja composto por quatro pontos: 1) o que está sendo proposto, 2) qual o mérito da proposta, 3) como a proposta foi avaliada/validada, 4) quais as possibilidades para trabalhos futuros. É seguido de (geralmente) três palavras-chave que devem indicar claramente a que se refere o seu trabalho. Por exemplo: *Este trabalho apresenta informações úteis a produção de trabalhos científicos para descrever e exemplificar como utililzar a classe LaTeX do Departamento de Ciência da Computação da Universidade de Brasília para gerar documentos. A classe `UnB-CIC` define um padrão de formato para textos do CIC, facilitando a geração de textos e permitindo que os autores foquem apenas no conteúdo. O formato foi aprovado pelos professores do Departamento e utilizado para gerar este documento. Melhorias futuras incluem manutenção contínua da classe e aprimoramento do texto explicativo.*

Palavras-chave: meta-aprendizado, aprendizado de máquina, mudança de conceito, fluxos de dados

Abstract

O *abstract* é o resumo feito na língua Inglesa. Embora o conteúdo apresentado deva ser o mesmo, este texto não deve ser a tradução literal de cada palavra ou frase do resumo, muito menos feito em um tradutor automático. É uma língua diferente e o texto deveria ser escrito de acordo com suas nuances (aproveite para ler `http://dx.doi.org/10.6061%2Fclinics%2F2014(03)01`). Por exemplo: *This work presents useful information on how to create a scientific text to describe and provide examples of how to use the Computer Science Department's LaTeX class. The `UnB-CIC` class defines a standard format for texts, simplifying the process of generating CIC documents and enabling authors to focus only on content. The standard was approved by the Department's professors and used to create this document. Future work includes continued support for the class and improvements on the explanatory text.*

Keywords: metalearning, machine learning, concept shift, datastreams

Sumário

1 Introdução **1**

2 Aprendizado de Máquina e Fluxos de Dados **4**
2.1 Aprendizado de Máquina 5
2.1.1 Árvores de Decisão 7
2.1.2 Floresta Aleatória 8
2.1.3 Máquina de Reforço do Gradiente 9
2.1.4 Máquina de Vetores de Suporte 9
2.1.5 "Não existe almoço grátis" 10
2.2 Meta-Aprendizado 11
2.3 Fluxos de Dados 12
2.4 Metastream 15
2.4.1 Fase Offline 16
2.4.2 Fase Online 16

3 Hipóteses e Objetivos **18**
3.1 Hipóteses 18
3.2 Objetivos 18

Referências **19**

Lista de Figuras

2.1 Exemplo de árvore de decisão 7
2.2 Máquina de vetores de suporte. A direita temos exemplos de dados que são linearmente separáveis logo é possível margens rigidas e a direita dados não-linearmente separáveis onde são necessárias margens permissivas. Os pontos ξ_i são aqueles que estão do lado errado do plano. 10
2.3 Exemplo de tendência. 13
2.4 Exemplo de sazonalidade. 14
2.5 Extração de meta-atributos da janela ω_b e obtenção de rotulo da janela η_b. . 16
2.6 Discretização de janelas no nível base do fluxo de dados. 17
2.7 Extração de meta-atributos das janelas ω_b e η_b no nível meta. 17

Capítulo 1

Introdução

A iteração de usuários com sistemas digitais tem crescido exponencialmente com o progresso tecnológico [1] e uma quantidade massiva de dados vem sendo produzida nos últimos anos [2]. Cliques em paginas de mídias sociais, interações de usuários em serviços de *streaming*, sensores ou dispositivos *smart*[1], são exemplos de tecnologias que geram dados a uma alta frequência em grande volume diariamente.

Essa grande quantidade de informação se tornou um dos bens mais valiosos do mundo moderno [3]. Na "era do Big Data" [4, 5], companhias como Google®, Facebook®, Netflix® e Amazon® tem tirado proveito dessas grandes quantidades de dados através da mineração com objetivo de agregar informações valiosas aos usuários desses sistemas, como recomendações de conteúdo (filmes, músicas, livros.) ou filtrar conteúdos customizados para cada usuário.

Grande parte dos sistemas que era baseado em uso de memoria RAM para o processamento se tornou inviável para essa industria que produz dados na escala de *Petabytes* [1], portanto, foi necessário a concepção de novos ambientes para operarem no armazenamento e processamento desses dados [6]. Ferramentas baseadas em computação distribuída se apresentaram como uma alternativa viável devido seu potencial de escalabilidade quase linear.

Dentre essas ferramentas podemos exemplificar o Apache Hadoop®, que fornece um ambiente de armazenamento distribuído, segmentando e replicando arquivos para que sejam distribuídos aos nós de um *cluster* de forma automática, o Apache Spark®, que provê um ambiente de computação distribuída, seguindo o paradigma MapReduce e o Apache Kafka® o qual provê um ambiente para lidar com fluxos de dados, o qual é o motivador desse trabalho.

Fluxos de dados, ou *Data Streams*, são conjuntos de dados gerados em tempo real de forma continua a frequências variáveis, estes atrelados aos pacotes de transmissão. Em-

[1] Também referidos como dispositivos *Internet of Things* (IoT)

bora fontes como essa gerem dados sem qualquer supervisão ou filtragem, essa quantidade massiva e complexa de dados tem uma capacidade surpreendente de se extrair informação de grande valor, como discutido em "*The Unreasonable Effectiveness of Data*" [2].

Uma das abordagens para extração de informação que tem sobressaído aliado ao *Big Data* é o Aprendizado de Máquina (AM). O AM é o campo da inteligência artificial que estuda métodos para a extração de padrões dos dados, para que esse conhecimento seja aplicado em tarefas futuras [7, 8].

Como todo método indutivo, algoritmos tradicionais de AM assumem duas premissas que devem ser satisfeitas para que funcionem corretamente: (i) o futuro deve se comportar como o passado e (ii) eventos futuros devem ser independentes de eventos passado [9]. Entretanto, em um fluxo de dados continuo (em inglês, *Data Stream*), a distribuição que gera os dados usualmente está mudando com o tempo e dependências temporais podem ocorrer. Logo, essas premissas podem não se manter e a performance dos modelos induzidos podem perecer com o passar do tempo, causando uma má experiência aos usuários desses sistemas [6, 10].

Diversos métodos foram desenvolvidos para lidar em ambientes com mudanças temporais, como detecção de mudança de conceito [11], retreino periódico de algoritmo [12] ou novas abordagens de algoritmos, especialmente projetados para esse contexto [13].

Por exemplo, alguns métodos de detecção de mudança de conceito [14] são baseados em alarmes que disparam quando a diferença da performance do modelo atual e do melhor modelo até então é maior que um dado limiar. Outros métodos, como a indução periódica de modelos, podem não ser suficientes para resolver esses problemas, já que o espaço de hipótese fixado do algoritmo pode não mais ser apropriado ao problema [15].

Meta-Aprendizado (MtA) é uma técnica proeminente para resolver essas limitações através da detecção de mudança de conceito e recomendação de algoritmos para melhorar a predição [16, 17, 18]. Para usar MtA, meta-dados devem ser construídos [19] baseado em uma conjunto de características descritivas, nomeados meta-atributos, do problema sob analise [20]. Esses meta-atributos são extraídos e combinados criando um meta-exemplo. Diferentes algoritmos de AM são aplicados ao problema, e suas performances são utilizadas para rotular os meta-exemplos. Esse procedimento é performado para o fluxo de dados em diferentes pontos no tempo, resultando em um conjunto de meta-dados, o qual pode ser usado para induzir um modelo capaz de predizer qual será o algoritmo mais apropriado para eventos futuros sem os altos custos usuais que o processo de seleção de modelos de ML demanda [21].

Nesse trabalho o MetaStream [22, 15], um *framework* baseado em seleção periódica de algoritmos em fluxos de dados usando MtA, é aprimorado pela extensão de meta-atributos para mais modernos e informativos [20], e incluindo aprendizado incremental no nível

meta, propondo o LightGBM [23] como meta-classificador, dado suas capacidades para lidar com esse problema específico. Seguimos então para investigar se meta-atributos de ponta e aprendizado incremental provido pelo LightGBM são capazes de predizer de forma mais precisa que o método base e aprimorar a performance geral do sistema aprendiz.

Capítulo 2

Aprendizado de Máquina e Fluxos de Dados

A necessidade de mecanizar tarefas humanas data desde o período clássico da civilização, em cerca de 350 a.C. na Grécia Antiga. Aristóteles, filosofo e aluno de Platão, entre diversos escritos que produziu, investigou o homem, a mente e a razão [24]. Em seu livro *Política*, um trabalho de ética e filosofia política, Aristóteles descreve a necessidade de automatizar tarefas humanas por mecanismo que obedeçam "palavras de comando ou antecipação inteligente" [25], como forma de resolver problemas sociais, como a escravidão. Não apenas nesse livro, mas outros trabalhos de seus trabalhos foram precursores [26] no desdobramento de um esforço coletivo para criar máquinas que poderiam reproduzir atividades humanas.

Uma das primeiras abordagens que poderia realizar tal feito surge muito antes, em cerca de 2500 a.C., onde os matemáticos babilônios descrevem uma serie de etapas e operações a se seguir para realizar a divisão entre dois números [27], tal procedimento foi posteriormente nomeado de 'algoritmo'.

Com o advento da computação e a concepção da máquina universal, o termo algoritmo foi adotado por esta ciência para designar instruções finitas e bem definidas que podem ser implementadas em um computador para resolver classes de problemas [28], tal abordagem remete aquela almejada por Aristóteles ao especificar "palavras de comando" que deveriam ser seguidas por um mecanismo autônomo.

Duas alternativas computacionais emergiram de forma promissoras para a concepção desses mecanismos, o raciocínio, através de algoritmos de dedução, e a busca, através de algoritmos que percorrem árvores de possibilidades. Porém tais métodos apresentam limitações teóricas e práticas, como descrito em [29] apenas o método indutivo é incapaz de criar novo conhecimento, necessitando de uma abordagem "criativa", e do ponto de vista prático o raciocínio dedutivo não se apresenta como uma alternativa razoável em um

cenário de incerteza [26], e os métodos de busca se tornam intratáveis se a complexidade do problema é muito alta [30, 31], é necessária então uma abordagem indutiva.

Presente em diversos organismos na natureza [32], a aprendizagem é uma das ferramentas cognitivas que permite não necessitar de informações completas e permite lidar com incerteza gerando soluções quase ótimas aos problemas. Um exemplo do processo de aprendizado é observado em ratos evitando iscas envenenadas. Num primeiro momento em que ratos encontram comidas com aspectos ou cheiros desconhecidos, eles inicialmente irão comer apenas pequenas quantidades, e dependendo do resultado dessa experiência, caso a comida produza más sensações, a comida será associada a algo ruim e subsequentemente será evitada, o processo chamado "aversão condicionada ao sabor" [33].

Tal motivações criaram a terceira abordagem para máquinas operarem tarefas humanas, o Aprendizado de Máquina, essa que se mostrou bem sucedida em tarefas que as outras abordagens não apresentavam desempenho tolerável. Na próxima seção definimos o aprendizado em seu aspecto computacional, diferente do aprendizado estatístico como em [8], este trabalho será baseado nas teorias computacionais de aprendizado [34, 35, 36] com o devido formalismo e evidenciando as limitações de computabilidade.

2.1 Aprendizado de Máquina

No contexto de inteligência artificial, o **Aprendizado**, ou Aprendizado de Máquina (AM), é o estudo e a concepção de agentes que melhoram o seu desempenho nas tarefas futuras de aprendizagem após fazer observações sobre o mundo [26, 37]. Em [7] é formalmente definido como:

> **Definição 1** Um programa de computador (agente), *aprende* de uma experiência $\boldsymbol{E}$ a respeito de alguma classe de tarefa $\boldsymbol{T}$ e medida de performance $\boldsymbol{P}$, se a performance na tarefa $\boldsymbol{T}$, medido por $\boldsymbol{P}$, melhora com a experiência $\boldsymbol{E}$.

No modelo **supervisionado**[1], o qual lidaremos exclusivamente neste trabalho, a experiência $\boldsymbol{E}$ é um registro $\boldsymbol{x} = (x_1, x_2, ..., x_n)$, em que x_i é um valor real computável ($\mathbb{R}_c$), e seu respectivo rotulo y. A tarefa $\boldsymbol{T}$ consiste em atribuir um valor $\hat{y}$ para um $\boldsymbol{x}$ onde o y real ainda não é conhecido, mas assumindo que esse par ordenado $(\boldsymbol{x}, y)$ será gerado por um programa desconhecido porém que assumimos sua invariância e regularidade de aprendizagem [37, 34].

Para isto, queremos induzir através dos pares de experiência $(\boldsymbol{x}, y)$ uma **função computável** f_c, isto é, uma máquina de Turing (MT) que sobre a entrada $\boldsymbol{w}$ para com

[1] A literatura subdivide problemas de aprendizado em supervisionado, não supervisionado e por reforço [26, 8, 38], porém há outros paradigmas mais gerais, como a identificação de linguagem no limite [36]

exatamente $f_c(\boldsymbol{w})$ sobre sua fita, de tal forma que ao mensurarmos observações futuras de $f_c(\boldsymbol{x})$ por $\boldsymbol{P}$, o valor de $\boldsymbol{P}(\hat{y}, y)$ tenda a um erro irredutível [37, 7, 8].

Codificaremos y em dois tipos de tarefas em $\boldsymbol{T}$, na primeira queremos computar uma cadeia de bits que representa um valor **real computável** a partir da entrada $\boldsymbol{x}$, ou seja, queremos uma f_c tal que $f_c : \mathbb{R}_c^n \longrightarrow \mathbb{R}_c$, essa tarefa denominamos **regressão**. Na segunda tarefa, queremos descobrir a função f que tenha como saída um único bit, de tal forma que $f : \mathbb{R}_c^n \longrightarrow \{0, 1\}$, essa tarefa denominamos **classificação**[2] [38, 37].

O processo indutivo descrito anteriormente aparenta ser uma tarefa fácil, porém para um número finitos de exemplos existem infinitas funções (f_c) que interpolam perfeitamente todos esses pontos[40, 39], entretanto, para que se tenha um bom desempenho em tarefas futuras, tal função deve se aproximar do real programa gerador dos dados [37, 38, 41, 42], o que nos leva a pergunta: "qual dessas funções deve ser escolhida?"

Um princípio que pode ser adotado nessa escolha, é o chamado *"Navalha de Ockham"* [43], ele diz que para duas hipóteses de igual poder explicativo, é preferível escolher a *mais simples*. No inicio do desenvolvimento de técnicas de inferência indutiva, não havia uma definição formal do que seria uma hipótese "mais simples", em 1964 com o trabalho de Solomonoff da teoria de inferência indutiva universal [41], é apresentada uma definição formal para o que seria esse principio: a MT de menor comprimento que computa tal observação, isso é, a f_c com menor complexidade de Kolmogorov. Esse sistema indutivo do ponto de vista teórico pode ser considerado perfeito [44], já que assumindo que o mundo é gerado por um programa desconhecido, na distribuição de probabilidade algorítmica, programas de menor descrição são os mais prováveis.

Porém já se sabe que tal MT (programa) é **indecidível**, pois escolher um programa equivalente de menor comprimento é mapeável ao problema da parada [45, 46], logo, no caso geral, seria impossível descrever um **método efetivo** para proceder tal escolha [47]. Uma solução a esse problema[3] é definir *apriori* um espaço de hipóteses $\mathcal{H}$ (programas) de menor cardinalidade[4], e.g. todos os programas que podem ser descritos em Python com no máximo 10^9 bits, assim, algoritmos efetivos poderiam escolher os programas de menor descrição em seu respectivo $\mathcal{H}$ [49].

Embora não exista uma máquina de Turing (MT) que resolva o caso geral, a teoria Provavelmente Aproximadamente Correto de aprendizado nos dá a garantias da existência de máquinas que resolvam em tempo polinomial, isto é, para alguma amostra de tamanho n, existe uma MT que com tempo $O(n^k)$ para algum $k \in \mathbb{N}$, encontra uma MT c que

[2]Embora definido para duas classes o formalismo descrito aqui pode ser estendido para mais de duas classes com métodos multi classe [39].

[3]As teorias Dimensão VC [9], Complexidade de Rademacher [35] e PAC [34] são casos restritos da inferência indutiva universal [44, 48]

[4]Também referido como Viés.

mapeia $c : \mathcal{X} \to \mathcal{Y}$, de tal forma que essa MT se aproxime da real função geradora f_c com alta probabilidade. Para isso a classe de programas geradores que se pode assumir deve ser limitada, apenas dados gerados de forma independente por uma distribuição estacionaria nos garante a validade das propriedades dos teoremas apresentados a seguir.

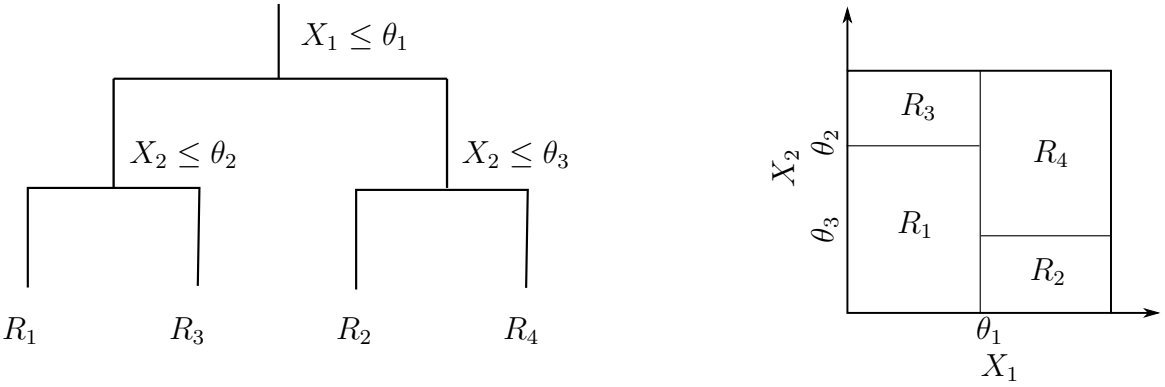
Limitemos as funções computáveis que queremos descobrir a retângulos que classificam pontos em $\mathbb{R}^2$, os quais pertencem aos conjunto de conceitos $\mathcal{C}$. Seja um retângulo c de tal forma que c classifique o ponto em 1 se contido dentro do retângulo ou 0 c.c, isto é, $c : \mathcal{X} \to \mathcal{Y}$, sendo $\mathcal{X} \in \mathbb{R}^2$, as coordenadas no plano cartesiano e $\mathcal{Y} \in \{0, 1\}$, a classe ao qual aquele ponto é atribuída. O algoritmo deve conter um espaço de hipóteses fixo $\mathcal{H}$ que não necessariamente coincida com $\mathcal{C}$ e obter amostras $S = (X_1, X_2, ..., X_m)$ obtidas da distribuição $\mathcal{D}$, esta i.i.d. (independente e identicamente distribuído) como descrito anteriormente.

falar do pac learning, np-complexidade

Existe uma diversidade de algoritmos para seleção de hipóteses [40, 8], neste trabalho apresentaremos a Árvore de Decisão, e algoritmos derivados como Floresta Aleatória e Máquinas de Reforço (*Boosting*) do Gradiente, também apresentaremos as Máquinas de Vetores de Suporte.

2.1.1 Árvores de Decisão

Árvores de Decisão são modelos que devido sua simplicidade tem fácil interpretabilidade e podem ser combinado com técnicas de amostragem e reforço em algoritmos em modelos muito mais poderosos. Embora exista um diversidade de algoritmos para sua concepção nos basearemos no modelo CART proposto em [50], a intuição por trás desse algoritmo é particionar o espaço de atributos $\mathcal{X}$ em regiões em R_i, de tal forma que cada R_i se aproxime do valor médio de $\mathcal{Y}$ para aquela região.



(a) Árvore de decisão com predicados booleanos sobre os atributos.

(b) Plano cartesiano particionado por uma árvore de decisão.

Figura 2.1: Exemplo de árvore de decisão

Na Figura 2.1b é exemplificado um particionamento dado pela árvore da Figura 2.1a, na raiz da árvore de decisão é aplicado um predicado booleano que irá dividir os dados em dois subconjuntos, a cada divisão nos nós esses subconjuntos são aplicados a novos predicados gerando subconjuntos menores.

Explicada o funcionamento das árvores, detalhamos agora como construí-las. Para isso, é feito uma busca exaustiva em todos os atributos j e seus valores t_m bi particionando os dados em $\theta = (j, t_m)$ como mostrado em 2.1.

$$\begin{aligned} Q_{\text{left}}(\theta) &= (x, y)|x_j <= t_m \\ Q_{\text{right}}(\theta) &= Q \setminus Q_{\text{left}}(\theta) \end{aligned} \tag{2.1}$$

De tal forma a minimizar a equação 2.2 sobre a função de impureza H.

$$\text{argmin}_\theta \frac{n_{\text{left}}}{N_m} H(Q_{\text{left}}(\theta)) + \frac{n_{\text{right}}}{N_m} H(Q_{\text{right}}(\theta)) \tag{2.2}$$

Para o calculo do H é dado pela proporção de classes na região R_m, dado N_m observações, em razão do erro de classificação.

$$p_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k) \tag{2.3}$$

Em [8] são apresentados alguns dos critérios que podem ser usados para computar a impureza da classificação, aqui aplicaremos o índice Gini.

$$H(X_m) = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{2.4}$$

Esse particionamento é então aplicado recursivamente até que seja atingido algum critério de regularização, por exemplo, o limite mínimo para m ou definido um valor minimo para H.

Embora árvores de decisão tenha diversas vantagens, elas tendem a sobreajustar aos dados, uma possível solução para isso é o uso de comitês.

2.1.2 Floresta Aleatória

Árvores de decisão podem "decorar" os dados de treino, não generalizando para casos desconhecidos, o chamado sobreajuste. O *Bagging*, uma técnica de reamostragem, tenta diminuir esse sobreajuste por meio da diminuição da variação entre os modelos, ou seja, modelos obtidos de forma independente tendam a uma mesma predição. Ele consiste em designar subamostras aleatórias (com repetição) do conjunto de dados e ao final as estimativas desses modelos são combinadas.

Porém ainda assim, dado a natureza determinística e a flexibilidade de uma árvore de decisão o problema do sobreajuste pode não ser resolvido, é necessário então aleatorizar o processo de geração da árvore. O método de subespaço aleatório (*random subspace method*) provê isso limitando o espaço de busca por bipartições da árvore, a cada novo nó da árvore que está sendo construída, apenas um subconjunto aleatório dos atributos pode ser escolhido para a bipartição dos dados.

Algorithm 1: Floresta Aleatória

Data: Conjunto de dados D com atributos p.

Input: B - Número de estimadores a serem construídos, m - Número de atributos nos subconjuntos.

Output: Classificador - Maioria dos votos de $\{T_b\}^B$.

1. Para b=1 até B.
   (a) Obtenha um subconjunto aleatório com repetição (*Bootstrap*) de D.
   (b) Construa uma árvore T_b recursivamente da seguinte forma:
      i. Selecione m variáveis do conjunto p.
      ii. Selecione a melhor variável e bipartição em m.
      iii. Divida o nó em dois nós filhos.
2. Retorne o conjunto de arvores $\{T_b\}^B$.

Essa limitação garante que as árvores de entropia máxima não possam ser sempre geradas, diminuindo a correlação entre as árvores e, portanto, melhorando a capacidade de generalização desse estimador. No Algoritmo 1 é apresentado o pseudo-código da Floresta Aleatória.

2.1.3 Máquina de Reforço do Gradiente

2.1.4 Máquina de Vetores de Suporte

Máquinas de vetores de suporte são construídas em cima de duas idéias muito elegantes do ponto de vista teórico, classificadores de máxima margem e projeções vetoriais. Classificadores de máxima margem obtém um hiperplano de solução unica tal que a margem entre esse hiperplano e os pontos de duas classes, se linearmente separáveis, seja máximo, resultando em um hiperplano com garantias de uma boa generalização [9].

Porém, uma vasta gama problemas não apresenta uma separação linear entre as classes, sendo necessárias duas abordagens generalizar esse algoritmo, projetar os dados em uma dimensão em que sejam linearmente separaveis, possívelmente uma dimensão infinita

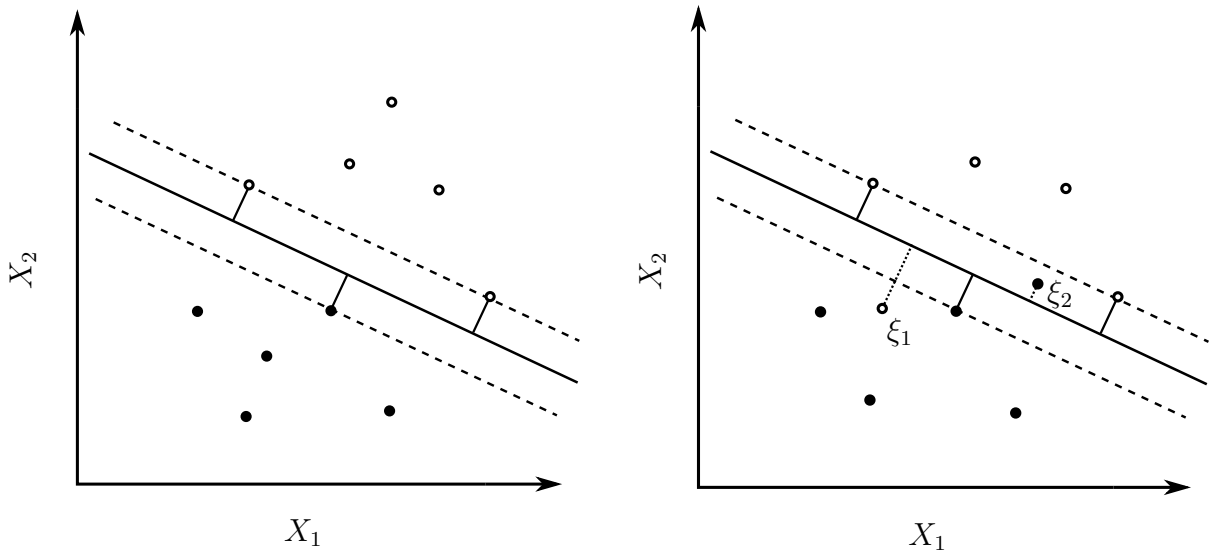


Figura 2.2: Máquina de vetores de suporte. A direita temos exemplos de dados que são linearmente separáveis logo é possível margens rigidas e a direita dados não-linearmente separáveis onde são necessárias margens permissivas. Os pontos ξ_i são aqueles que estão do lado errado do plano.

através do *kernel trick*, e também tornar as margens menos restritas para se adaptarem a ruídos nos dados. Inicialmente definimos o hiperplano como

$$x^T\beta + \beta_0 = 0 \tag{2.5}$$

Sendo β o vetor unitário, isto é $\|\beta\| = 1$ e a regra de classificação é dada por

$$G(x) = \text{sign}(x^T\beta + \beta_0) \tag{2.6}$$

Para obter o hiperplano de máxima margem otimizamos β restringindo o critério de otimização de tal forma que suavize para os pontos ξ_i, logo

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i \quad \text{subject to } \xi_i \geq 0, y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i \forall i \tag{2.7}$$

Onde C é um hiperparametro definido *apriori*.

descrever kernels

2.1.5 "Não existe almoço grátis"

Vimos a viabilidade e descrevemos o projeto de alguns algoritmos, surge então a seguinte questão: "E se colocássemos apenas os 'bons programas', isto é, de alguma forma adicionar apenas aqueles de menor descrição, no espaço espaço de hipótese do algoritmo para ser

selecionado?" Devido ao teorema do *No Free Lunch* [51, 52] temos uma prova que tal algoritmo seria inconcebível, pois não é possível ter um melhor algoritmo para todos os casos [49], o que nos é apresentado intuitivamente pela teoria da inferência indutiva universal.

O algoritmo, e consequentemente o espaço de hipóteses $\mathcal{H}$, é usualmente selecionado com base em resultados empíricos. Tais avaliações são feitas baseadas métodos de validação, porém a escassez e o "vazamento" de dados costuma gerar o problema de sobreajuste, sendo então uma solução limitada ao problema. Há uma alternativa não baseada em avaliação empírica para a seleção de algoritmos, através do meta-aprendizado.

2.2 Meta-Aprendizado

O problema de seleção de algoritmos foi inicialmente observado por J. R. Rice (1976) [53] com o principal objetivo de prever o melhor algoritmo para resolver um dado problema quando houver mais de um algoritmo que resolva disponível. Os componentes desse modelo são: o espaço de instâncias do problema (P), que é composto por conjuntos de dados em Meta-Aprendizado (MtA); o espaço de instâncias de atributos (F), que são os meta-atributos usados para descrever os conjuntos de dados; o espaço de algoritmos (A), que contém um conjunto de algoritmos de AM que podem ser recomendados; e um espaço de medidas de avaliação (Y), responsável por recuperar as performances dos algoritmos de AM que resolvem as instâncias dos problemas contidos em P. Usando os conjuntos descritos anteriormente, um sistema de MtA pode criar o mapeamento de um conjunto de dados p descriminâveis pelos meta-atributos f em um ou mais algoritmos α, de tal forma que a recomendação de algoritmos por sistema tenha uma performance medida por y tolerável, e.g., com máximo $y(\alpha(p))$.

K. A. Smith-Miles (2008) [54] aprimorou esse modelo abstrato propondo generalizações que podem também ser aplicadas ao problema do projeto de algoritmos. Nessa proposta, alguns componentes são adicionados: o conjunto de algoritmos de MtA; a generalização de regras empíricas ou ranqueamento de algoritmos; a verificação de resultados empíricos, que podem ser guiados por uma base teórica ao aprimoramento de algoritmos.

Um componente crucial dos modelos anteriores é a definição do conjunto de meta-atributos (F) usados para descrever propriedades gerais dos conjuntos de dados. Esses meta-atributos devem ser capaz de prover evidências sobre performance futuro dos algoritmos em A [55, 56] e discriminar, com baixo custo computacional, a performance de um grupo de algoritmos. Os principais meta-atributos usados na literatura de MtA podem ser divididos em cinco grupos: Gerais, Estatísticos, de Teoria da Informação, Baseados em Modelo e de Landmarkings.

Os Gerais podem ser facilmente extraídos dos dados [57], com baixo custo computacional [56]. Os meta-atributos estatísticos capturam os indicadores principais sobre localização e distribuição dos dados, tais como média, desvio padrão, correlação e curtose. Meta-atributos de teoria da informação, usualmente medidas de entropia [58], capturam a quantidade de informação em um (sub)conjunto dos dados [54]. Já os baseados em modelo são propriedades extraídas de modelos de AM, geralmente AD [59, 60], induzidos dos dados sob analise [57]. Os meta-atributos de Landmarking usam a performance de algoritmos de aprendizado simples e rápidos para caracterizar os conjuntos de dados [54].

A definição do conjunto de instâncias de problema (P) é outra preocupação, quando o ideal seria usar um grande número de conjuntos de dados diversos, com objetivo de induzir um meta-modelo confiável. Para reduzir o viés dessa escolha, conjuntos de dados de diversos repositórios, tais como o UCI[5] [61] e o OpenML[6] [62], podem ser usados.

O espaço de algoritmos A representa o conjunto de algoritmos candidatos a serem recomendados no processo de seleção de algoritmos. Idealmente, esses algoritmos devem também ser suficientemente distintos entre si e representar toda a região do espaço de algoritmos [21]. Os modelos induzidos pelo algoritmo podem ser avaliados por diferentes medidas, para tarefas de classificação, a maioria dos estudos usando MtA usam acurácia. Entretanto, outros indicadores, como o F_β, AUC e coeficiente Kappa, também podem ser usados.

Após a extração dos meta-atributos dos conjuntos de dados e a mensuração da performance do conjunto de algoritmos nesses conjuntos de dados, o próximo passo é rotular cada meta-exemplo nos meta-dados. Brazdil et al. (2009) [63] resume as quatro propriedades principais frequentemente usadas para rotular meta-exemplos em MtA: (i) o algoritmo que apresenta a melhor performance no conjunto de dados (uma tarefa de classificação); (ii) o ranqueamento dos algoritmos de acordo com suas performances no conjunto de dados (uma tarefa de ranqueamento), onde o algoritmo com melhor performance está no topo do ranque; (iii) o valor de performance obtido por cada algoritmo individualmente no conjunto de dados (uma tarefa de regressão) e; (iv) a descrição do modelo, que é geralmente baseado em agrupamentos ou regras de associação.

2.3 Fluxos de Dados

Dados estáticos e bem estruturados não se apresentam como uma boa alternativa a modelagem em ambientes com massiva interação livre de usuários, para esses ambientes uma das possíveis abordagens são fluxos de dados[64]. Embora fluxos de dados sejam observa-

[5] `https://archive.ics.uci.edu/ml/index.php`
[6] `http://www.openml.org/`

dos no tempo, similar a definição de uma série temporal eles diferem de séries temporais em alguns aspectos, fluxos de dados não tem uma frequência fixa entre as observações, sendo essa frequência definida geralmente pela ordem aleatória da geração dos dados que pode ou não ter uma natureza evolutiva [6]. Outro aspecto importante ao lidar com desse tipo de dado são as limitações de engenharia, dado que é um conjunto infinito (não terminável) de dados e que não podem ser lidos livremente, dado os limites computacionais impostos pela grande quantidade de dados [64]. Por ultimo, acredita-se que fluxos de dados não tenham dependências temporais, diferentes de séries temporais, porém em artigos como [65] é apresentado que não é isso que ocorre no caso de mudança de conceito.

Essas limitações práticas causam limitações significativa nos métodos e abordagens que podem ser aplicados na mineração de fluxos de dados, essa que será tratada nessa seção. Séries temporais (e similarmente, fluxos de dados) podem ser gerados por dois tipos de distribuições, estacionárias e não-estacionárias, para a primeira se tem medias e variâncias constantes para qualquer janela ao longo do tempo, já na segunda, esses dois parâmetros podem variar em função de fenômenos conhecidos ou desconhecidos por trás do processo gerador desses dados[65]. Tais padrões de variações costumam recorrer em diversas series por isso nomes foram atribuídos a cada comportamento, são eles, tendência, ciclos, sazonalidade e mudança de conceito [66, 67, 68].

Tendências e ciclos geralmente descrevem eventos econômicos de crescimento (ou decrescimento) e oscilações de subida e descida dado por flutuações de curto prazo, para simplicidade e por esses fenômenos terem uma natureza similar, eles são agregados em uma única componente e denominados por "ciclo-tendência" ou apenas "tendência" [66].

Com o passar dos anos é esperado um crescimento populacional e naturalmente a população empregada também irá crescer, a Figura 2.3 apresenta a série temporal do número de pessoas formalmente empregadas nos Estados Unidos, do ano 1947 a 1962, é observável o efeito da tendência no comportamento da série temporal já que vemos o crescimento como descrito acima.

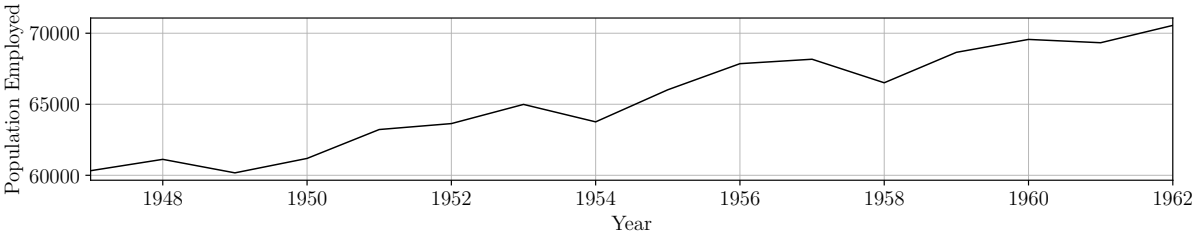


Figura 2.3: Exemplo de tendência.

Padrões sazonais ocorrem quando um fenômeno observado é afetado por fatores sazonais, por exemplo o período do ano ou a hora do dia, esse necessariamente de uma

frequência fixa conhecida [66, 68]. Na Figura 2.4, é apresentado uma série de observações da temperatura média diária na cidade de Melbourne, Austrália ao longo de 10 anos, essa séria apresenta um padrão de sazonalidade, sendo a temperatura da terra afetada pelo ciclos solar, as estações do ano, é possível observar a periodicidade do aumento e diminuição da temperatura relacionado a isso.

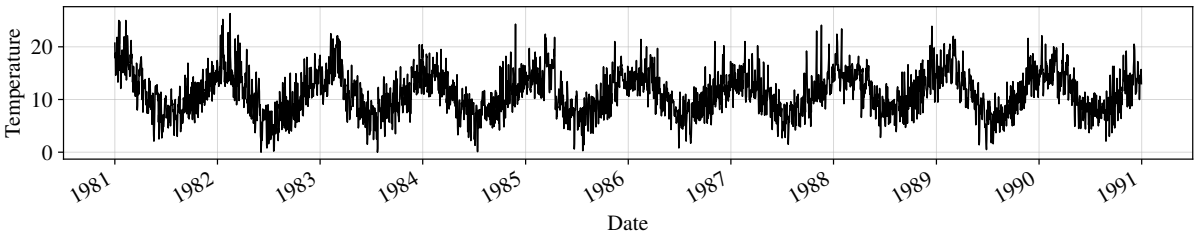


Figura 2.4: Exemplo de sazonalidade.

descrever mudança de conceito estatisticamente, detalhar

A maioria das técnicas de AM assumem que os dados são independentes e identicamente distribuídos (*iid*), mas essas premissas geralmente não se sustentam em ambientes de fluxos de dados, como apontado por A. Bifet and R. Gavaldà (2007) [12], dado que os dados chegam interminavelmente e, por isso, podem apresentar dependência temporal e mudanças de conceito pode ocorrer. Outro aspecto critico é a imposição de limites computacionais a esses ambientes, como memória, CPU, e largura de banda [69, 70]. Para resolver esses problemas, diversas tecnologias foram desenvolvidas. Dentre elas, as mais proeminentes são mecanismos de esquecimento, testes estatísticos e algoritmos adaptativos.

Mecanismos de esquecimento fazem dados recentes tornarem-se mais relevantes e dados passados menos, um exemplo são janelas deslizantes [71], um método agnóstico de modelo. Sistemas baseados em testes estatísticos agem monitorando uma métrica pre-definida, tal como performance preditiva do modelo, e então "disparam" um alarme quando a qualidade dessas métricas está abaixa de algum limiar tolerável, demandando alguma ação.

Exemplos dessa abordagem são o teste de Page-Hinkley e o *Statistical Process Control* [14]. Baseado nessas duas soluções, o *ADaptive WINdowing* (ADWIN) [12] usa testes estatísticos para definir o tamanho de cada janela, ajustando ao "tamanho do conceito". Entretanto, embora esses métodos serem capazes de alarmar uma redução na performance dos modelos e adaptar o tamanho da janela a isso, não é possível identificar os algoritmos ou modelos que se tornaram mais apropriados ao novo conjunto de dados que está sendo gerado.

Algoritmos de aprendizado adaptativo, como o VFDT [72], foram inicialmente introduzidos com fluxos de dados em mente, desempenhando aprendizado online e baixo consumo

de memoria, mas não conseguiam se adaptar a mudança de conceito, um problema já conhecido na época. Mais tarde, aprimoramentos foram propostos, como o CVFDT (um aprimoramento direto sobre o VFDT) [73], o HAT [74] e o ARF [75], que podem lidar com variação de conceito aplicando janelas deslizantes sobre o processo de treinamento. Entretanto, se a mudança for muito abrupta, o espaço de hipótese, fixado a priori do algoritmo (Arvore de Decisão (AD) para a maioria deles) pode não ser mais adequado e as adaptações desejadas não serem atingidas.

Outra alternativa são algoritmos genéticos, como em [76] e [77] e outros artigos usando meta-learning Esses problemas podem ser reduzidos quando se usa detectores de mudança baseado em sistemas de recomendação baseados em MtA, almejando espaços de hipótese como tarefas de aprendizado [15].

falar do [78]

2.4 Metastream

Em ambientes de fluxos de dados, o espaço de instância de problemas é composto por apenas um problema p enquanto os meta-atributos F são extraídos dos dados de cada janela temporal para compor o meta-exemplo. falar sobre custo O(N2) mas no caso de fluxos de dados deve ser um tempo menor do que o processamento de um batch da fila pois se não fila cresceria infinitamente É importante que o processo de extração dos meta-atributos tenha baixo custo computacional e alto grau de informação. Além disso, a performance desses algoritmos A são obtidas periodicamente, usualmente para cada janela deslizante. Portanto, o algoritmo atual é substituído logo que o meta-modelo prediz que um algoritmo diferente é mais adequado para os exemplos da próxima janela deslizante. Essa é uma abordagem padrão na literatura de MtA em fluxos de dados [79, 80, 16].

A literatura apresenta alternativas a seleção de algoritmos (ou modelos) baseado em MtA para mudança de conceito. Um dos primeiros a apresentar é R. Klinkenberg (2005) [81], onde características obtidas do processo de aprendizado em si, no caso o tamanho das janelas deslizantes, foram usadas para induzir meta-modelos. Uma abordagem diferente foi investigada por J. Gama and P. Kosina (2014) [82] e R. Anderson et al. (2019) [16] reusando modelos previamente aprendidos mas usando o mesmo algoritmo para induzir esses modelos no fluxo de dados. Uma terceira alternativa é dada por J. N. van Rijn et al. (2018) [83], onde é usado conjuntos de modelos, tendo um alto custo computacional para induzir modelos para cada algoritmo e manter os pesos atualizados.

Em seguida, é apresentado o MetaStream baseado no trabalho de A. L. D. Rossi et al. (2014) [15], que tem fases “offline” e “online”. A fase offline is projetada para escolha de hiper-parâmetros, validação e treinamento e geração dos dados em lote. A fase online

age no ambiente dinâmico, recomendando um algoritmo para uma dada janela de dados. Ambas as fases são baseadas em sistemas de recomendação de MtA.

2.4.1 Fase Offline

Essa fase se inicia aguardando o fluxo de dados gerar um lote de tamanho razoável para indução e seleção de modelos base, com esse lote em mãos, é realizada uma validação cruzada *k-fold* para estimar os hiper-parâmetros dos algoritmos base, aqui assumindo que tais hiper-parâmetros serão os melhores para os eventos futuros. Após isso, em um configuração de janelas deslizantes, como mostrado na Figura 2.5, uma janela ω_{b1} é usada para induzir modelos com os algoritmos de nível base e também os meta-atributos x^m também são extraídos dessa janela.

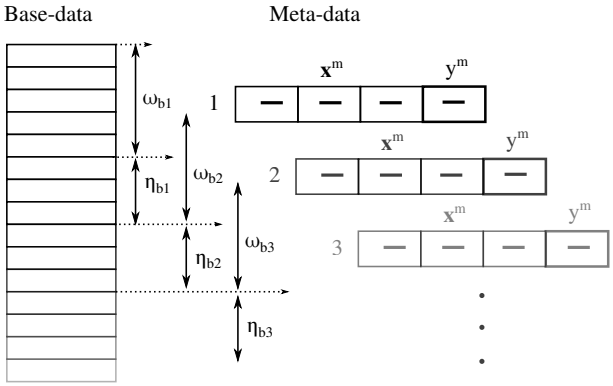


Figura 2.5: Extração de meta-atributos da janela ω_b e obtenção de rotulo da janela η_b.

Em seguida, os modelos induzidos são avaliados nos exemplos da janela η_{b1}, onde o algoritmo que produziu o modelo de melhor performance preditiva é definido como rótulo de y^m. Esse processo gera o primeiro meta-exemplo. O mesmo processo é aplicado continuamente por N passos, onde N é o número mínimo de instâncias para induzir um modelo consistente para gerar os meta-dados iniciais. Esses N meta-exemplos serão os primeiros dados que serão usados para induzir modelos na fase online.

2.4.2 Fase Online

Na fase online, o *framework* recebendo um contínuo fluxo de dados. Inicialmente, ele recebe um vetor de atributos $\boldsymbol{x}_b = (x_1, ..., x_p)$, e após algum atraso, o atributo rótulo $y_b \in \{0, 1, .., k\}$ para classificação, onde k é o número de classes ou $y_b \in \mathbb{R}$ para regressão.

A Figura 2.6 mostra um dado momento t na fase online. Ele tem uma janela de tamanho fixo ω_b que será usado para induzir um modelo, uma janela de tamanho fixo η_b onde o modelo induzido em ω_b será avaliado assim que o rótulo referente for descoberto,

um tamanho γ_b que é o atraso em observar o rótulo para os exemplos na janela η_b e uma janela de tamanho variável λ_b de exemplos aguardando serem processados. Quando γ_b tem o mesmo tamanho que η_b, isto é, todos os rótulos de η_b foram obtidos, a janela ω_b é deslizada η_b instâncias para a direita e um novo modelo é induzido nessa janela.

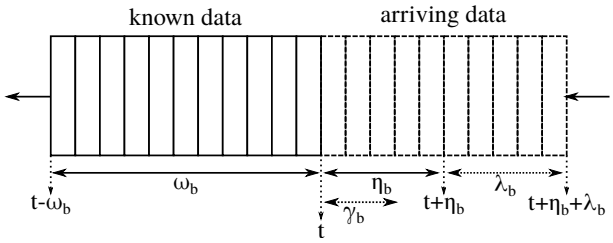


Figura 2.6: Discretização de janelas no nível base do fluxo de dados.

MtA entra em ação no segundo nível de processamento, nomeado nível meta. Como mostrado na Figura 2.7, os meta-atributos são extraídos da janela ω_b e geram o meta-exemplo sem o rótulo que é mantido para atualização posterior.

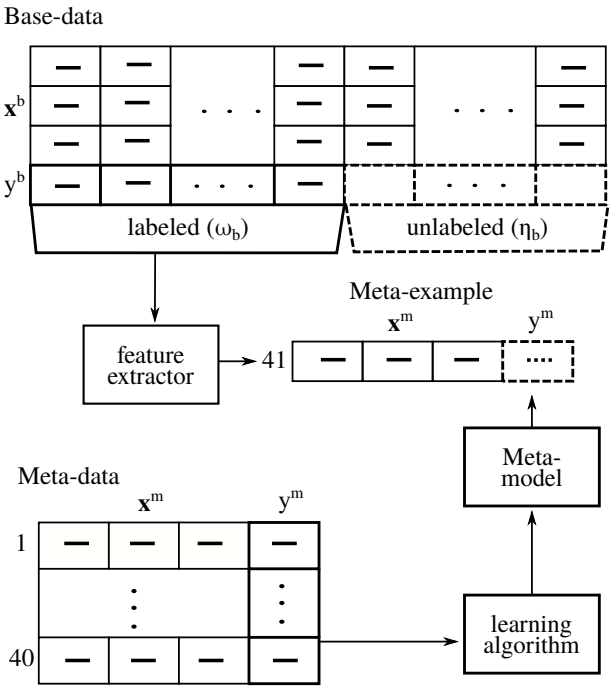


Figura 2.7: Extração de meta-atributos das janelas ω_b e η_b no nível meta.

O algoritmo de aprendizado (meta-aprendiz) usa meta-exemplos rotulados previamente na meta-base para induzir um meta-classificador, que é utilizado para recomendar um algoritmo que induzirá um modelo usando a janela ω_b que provavelmente será o mais preciso para prever os rótulos dos exemplos em η_b.

Capítulo 3

Hipóteses e Objetivos

3.1 Hipóteses

1. Em ambientes com mudança de conceito, viéses em um dado momento podem não ser adequados em momentos futuros.
2. Algoritmos incrementais aprensentam um grau elevado de regularização, isso pode ser benéfico para alguns ambientes.
3. Meta-Atributos modernos apresentam melhor poder descritivo, possibilitando uma melhora significativa em relação ao framework proposto originalmente.
4.

3.2 Objetivos

1. Apresentar ganhos preditivos e em performance em relação ao framework original.
2. Recomendar positivamente algoritmos de melhor performance ao longo do tempo.
3.

Referências

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

[2] Halevy, Alon, Peter Norvig e Fernando Pereira: *The unreasonable effectiveness of data*. IEEE Intelligent Systems, 24(2):8–12, 2009. 1, 2

[3] Economist, The: *The world's most valuable resource is no longer oil, but data*. The Economist: New York, NY, USA, 2017. `https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data`, acesso em 2020-03-14. 1

[4] Tarnoff, Ben: *Big data for the people: it's time to take it back from our tech overlords*, 2018. `https://www.theguardian.com/technology/2018/mar/14/tech-big-data-capitalism-give-wealth-back-to-people`, acesso em 2020-03-14. 1

[5] Finger, Lutz: *3 data products you need to know*, 2014. `https://www.forbes.com/sites/lutzfinger/2014/08/19/3-data-products-you-need-to-know`, acesso em 2020-03-14. 1

[6] Gama, João e Mohamed Medhat Gaber: *Learning from data streams: processing techniques in sensor networks*. Springer, 2007. 1, 2, 13

[7] Mitchell, Thomas M *et al.*: *Machine learning*. McGraw Hill series in computer science. McGraw-Hill, 1997. 2, 5, 6

[8] Friedman, Jerome, Trevor Hastie e Robert Tibshirani: *The elements of statistical learning*. Springer series in statistics New York, 2001. 2, 5, 6, 7, 8

[9] Vapnik, Vladimir: *The nature of statistical learning theory*. Springer science & business media, 2013. 2, 6, 9

[10] Johansson, Ulf, Cecilia Sönströd, Henrik Linusson e Henrik Boström: *Regression trees for streaming data with local performance guarantees*. Em *IEEE International Conference on Big Data (Big Data)*, páginas 461–470, 2014. 2

[11] Klinkenberg, Ralf e Thorsten Joachims: *Detecting concept drift with support vector machines*. Em *17th International Conference on Machine Learning (ICML)*, páginas 487–494, 2000. 2

[12] Bifet, Albert e Ricard Gavaldà: *Learning from time-changing data with adaptive windowing*. Em *International Conference on Data Mining (SIAM)*, páginas 443–448, 2007. 2, 14

[13] Zang, Wenyu, Peng Zhang, Chuan Zhou e Li Guo: *Comparative study between incremental and ensemble learning on data streams: Case study*. Journal Of Big Data, 1(5):1–16, 2014. 2

[14] Gama, João: *Knowledge discovery from data streams*. CRC Press, 2010. 2, 14

[15] Rossi, André L. D., André C. P. L. F. de Carvalho, Carlos Soares e Bruno Feres de Souza: *MetaStream: A meta-learning based method for periodic algorithm selection in time-changing data*. Neurocomputing, 127:52–64, 2014. 2, 15

[16] Anderson, Robert, Yun Sing Koh, Gillian Dobbie e Albert Bifet: *Recurring concept meta-learning for evolving data streams*. Expert Systems with Applications, 138:1–18, 2019. 2, 15

[17] van Rijn, Jan N., Geoffrey Holmes, Bernhard Pfahringer e Joaquin Vanschoren: *Having a blast: Meta-learning and heterogeneous ensembles for data streams*. Em *IEEE International Conference on Data Mining (ICDM)*, páginas 1003–1008, 2016. 2

[18] Zarmehri, Mohammad Nozari e Carlos Soares: *Using metalearning for prediction of taxi trip duration using different granularity levels*. Em *14th International Symposium on Intelligent Data Analysis (IDA)*, páginas 205–216, 2015. 2

[19] Vanschoren, Joaquin: *Meta-learning: A survey*. eprint arXiv, 1(1810.03548):1–29, 2018. 2

[20] Rivolli, Adriano, Luís P. F. Garcia, Carlos Soares, Joaquin Vanschoren e André C. P. L. F. de Carvalho: *Towards reproducible empirical research in meta-learning*. eprint arXiv, 1(1808.10406):1–41, 2018. 2

[21] Muñoz, Mario A, Laura Villanova, Davaatseren Baatar e Kate Smith-Miles: *Instance spaces for machine learning classification*. Machine Learning, 107(1):109–147, 2018. 2, 12

[22] Rossi, André L. D., Andre C. P. L. F. Carvalho e Carlos Soares: *Meta-learning for periodic algorithm selection in time-changing data*. Em *Brazilian Symposium on Neural Networks (SBRN)*, páginas 7–12, 2012. 2

[23] Ke, Guolin, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye e Tie Yan Liu: *LightGBM: A highly efficient gradient boosting decision tree*. Em *31st International Conference on Neural Information Processing System (NIPS)*, páginas 3149–3157, 2017. 3

[24] Russell, Bertrand: *History of western philosophy*. Routledge, 2004. 4

[25] Aristotle e C. Lord: *Aristotle's "Politics": Second Edition*. University of Chicago Press, 2013, ISBN 9780226921853. `https://books.google.com.br/books?id=DJP44GomyNoC`. 4

[26] Russell, Stuart e Peter Norvig: *Artificial intelligence: a modern approach*. Elsevier, 2016. 4, 5

[27] Chabert, Jean Luc, Évelyne Barbin *et al.*: *A history of algorithms: from the pebble to the microchip*, volume 23. Springer, 1999. 4

[28] Encyclopaedia Britannica, The Editors of: *Algorithm*. Encyclopædia Britannica, inc., 2006. `https://www.britannica.com/science/algorithm`. 4

[29] Popper, Karl: *The logic of scientific discovery*. Routledge, 2005. 4

[30] OpenAI, :, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski e Susan Zhang: *Dota 2 with large scale deep reinforcement learning*, 2019. 5

[31] Silver, David, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot *et al.*: *Mastering the game of go with deep neural networks and tree search*. nature, 529(7587):484, 2016. 5

[32] Shettleworth, Sara J: *Animal cognition and animal behaviour*. Animal behaviour, 61(2):277–286, 2001. 5

[33] Garcia, John, Donald J Kimeldorf e Robert A Koelling: *Conditioned aversion to saccharin resulting from exposure to gamma radiation*. Science, 1955. 5

[34] Valiant, Leslie G: *A theory of the learnable*. Communications of the ACM, 27(11):1134–1142, 1984. 5, 6

[35] Bartlett, Peter L e Shahar Mendelson: *Rademacher and gaussian complexities: Risk bounds and structural results*. Journal of Machine Learning Research, 3(Nov):463–482, 2002. 5, 6

[36] Gold, E Mark: *Language identification in the limit*. Information and control, 10(5):447–474, 1967. 5

[37] Mohri, Mehryar, Afshin Rostamizadeh e Ameet Talwalkar: *Foundations of machine learning*. MIT press, 2018. 5, 6

[38] Goodfellow, Ian, Yoshua Bengio e Aaron Courville: *Deep learning*. MIT press, 2016. 5, 6

[39] Abu-Mostafa, Yaser S, Malik Magdon-Ismail e Hsuan Tien Lin: *Learning from data*, volume 4. AMLBook New York, NY, USA:, 2012. 6

[40] Bishop, Christopher M: *Pattern recognition and machine learning*. springer, 2006. 6, 7

[41] Solomonoff, Ray J: *A formal theory of inductive inference. part i.* Information and control, 7(1):1–22, 1964. 6

[42] Kearns, Michael J e Umesh Vazirani: *An introduction to computational learning theory.* MIT press, 1994. 6

[43] Blumer, Anselm, Andrzej Ehrenfeucht, David Haussler e Manfred K Warmuth: *Occam's razor.* Information processing letters, 24(6):377–380, 1987. 6

[44] Li, Ming e Paul MB Vitányi: *Inductive reasoning and kolmogorov complexity.* Journal of Computer and System Sciences, 44(2):343–384, 1992. 6

[45] Sipser, Michael: *Introduction to the Theory of Computation.* Cengage Learning, 2012. 6

[46] Solomonoff, Ray: *Complexity-based induction systems: comparisons and convergence theorems.* IEEE transactions on Information Theory, 24(4):422–432, 1978. 6

[47] Hutter, Marcus: *Universal artificial intelligence: Sequential decisions based on algorithmic probability.* Springer Science & Business Media, 2004. 6

[48] Blumer, Anselm, Andrzej Ehrenfeucht, David Haussler e Manfred K Warmuth: *Learnability and the vapnik-chervonenkis dimension.* Journal of the ACM (JACM), 36(4):929–965, 1989. 6

[49] Rathmanner, Samuel e Marcus Hutter: *A philosophical treatise of universal induction.* CoRR, abs/1105.5721, 2011. `http://arxiv.org/abs/1105.5721`. 6, 11

[50] Breiman, Leo, Jerome Friedman, Charles J Stone e Richard A Olshen: *Classification and regression trees.* CRC press, 1984. 7

[51] Wolpert, David H e William G Macready: *No free lunch theorems for optimization.* IEEE transactions on evolutionary computation, 1(1):67–82, 1997. 11

[52] Wolpert, David H: *The lack of a priori distinctions between learning algorithms.* Neural computation, 8(7):1341–1390, 1996. 11

[53] Rice, John R.: *The algorithm selection problem.* Advances in Computers, 15:65–118, 1976. 11

[54] Smith-Miles, Kate A.: *Cross-disciplinary perspectives on meta-learning for algorithm selection.* ACM Computing Surveys, 41(1):1–25, 2008. 11, 12

[55] Soares, Carlos, Johann Petrak e Pavel Brazdil: *Sampling-based relative landmarks: Systematically test-driving algorithms before choosing.* Em *10th Portuguese Conference on Artificial Intelligence (EPIA)*, páginas 88–95, 2001. 11

[56] Reif, Matthias: *A comprehensive dataset for evaluating approaches of various meta-learning tasks.* Em *1st International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, páginas 273–276, 2012. 11, 12

[57] Reif, Matthias, Faisal Shafait, Markus Goldstein, Thomas Breuel e Andreas Dengel: *Automatic classifier selection for non-experts*. Pattern Analysis and Applications, 17(1):83–96, 2014. 12

[58] Segrera, Saddys, Joel Pinho e María N. Moreno: *Information-theoretic measures for meta-learning*. Em *3rd Hybrid Artificial Intelligence Systems (HAIS)*, páginas 458–465, 2008. 12

[59] Bensusan, Hilan, Christophe Giraud-Carrier e Claire Kennedy: *A higher-order approach to meta-learning*. Em *10th International Conference Inductive Logic Programming (ILP)*, páginas 1–10, 2000. 12

[60] Peng, Yonghong, Peter A. Flach, Carlos Soares e Pavel Brazdil: *Improved dataset characterisation for meta-learning*. Em *5th International Conference on Discovery Science (DS)*, volume 2534, páginas 141–152, 2002. 12

[61] Dua, Dheeru e Casey Graff: *UCI machine learning repository*, 2017. `http://archive.ics.uci.edu/ml`. 12

[62] Vanschoren, Joaquin, Jan N. van Rijn, Bernd Bischl e Luis Torgo: *OpenML: networked science in machine learning*. SIGKDD Explorations, 15(2):49–60, 2013. 12

[63] Brazdil, Pavel, Christophe Giraud-Carrier, Carlos Soares e Ricardo Vilalta: *Metalearning - Applications to Data Mining*. Cognitive Technologies. Springer, 1ª edição, 2009. 12

[64] Babcock, Brian, Shivnath Babu, Mayur Datar, Rajeev Motwani e Jennifer Widom: *Models and issues in data stream systems*. Em *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, páginas 1–16, 2002. 12, 13

[65] Read, Jesse: *Concept-drifting data streams are time series; the case for continuous adaptation*. arXiv preprint arXiv:1810.02266, 2018. 13

[66] Hyndman, Rob J. e George Athanasopoulos: *Forecasting: principles and practice*. OTexts, 2018. 13, 14

[67] Tsymbal, Alexey: *The problem of concept drift: definitions and related work*. Computer Science Department, Trinity College Dublin, 106(2):58, 2004. 13

[68] Brockwell, Peter J e Richard A Davis: *Introduction to time series and forecasting*. springer, 2016. 13, 14

[69] Bifet, Albert, Geoff Holmes, Richard Kirkby e Bernhard Pfahringer: *MOA: Massive Online Analysis*. Journal of Machine Learning Research, 11:1601–1604, 2010. 14

[70] Gama, João: *A survey on learning from data streams: current and future trends*. Progress in Artificial Intelligence, 1(1):45–55, 2012. 14

[71] Gaber, Mohamed Medhat, Arkady Zaslavsky e Shonali Krishnaswamy: *Mining data streams: a review*. ACM SIGMOD Record, 34(2):18–26, 2005. 14

[72] Domingos, Pedro e Geoff Hulten: *Mining high-speed data streams.* Em *6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, páginas 71–80, 2000. 14

[73] Hulten, Geoff, Laurie Spencer e Pedro Domingos: *Mining time-changing data streams.* Em *7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, páginas 97–106, 2001. 15

[74] Bifet, Albert e Ricard Gavaldà: *Adaptive learning from evolving data streams.* Em *8th International Symposium on Intelligent Data Analysis (IDA)*, páginas 249–260, 2009. 15

[75] Gomes, Heitor M, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes e Talel Abdessalem: *Adaptive random forests for evolving data stream classification.* Machine Learning, 106(9-10):1469–1495, 2017. 15

[76] Kanade, Varun, Leslie G. Valiant e Jennifer Wortman Vaughan: *Evolution with drifting targets*, 2010. 15

[77] Kang, Yanfei, Rob J Hyndman e Kate Smith-Miles: *Visualising forecasting algorithm performance using time series instance spaces.* International Journal of Forecasting, 33(2):345–358, 2017. 15

[78] Talagala, Thiyanga S, Rob J Hyndman, George Athanasopoulos *et al.*: *Meta-learning how to forecast time series.* Monash Econometrics and Business Statistics Working Papers, 6:18, 2018. 15

[79] Read, Jesse, Albert Bifet, Bernhard Pfahringer e Geoff Holmes: *Batch-incremental versus instance-incremental learning in dynamic and evolving data.* Em *11th International Symposium on Intelligent Data Analysis (IDA)*, páginas 313–323, 2012. 15

[80] van Rijn, Jan N., Geoffrey Holmes, Bernhard Pfahringer e Joaquin Vanschoren: *Algorithm selection on data streams.* Em *17th International Conference on Discovery Science (DS)*, páginas 325–336, 2014. 15

[81] Klinkenberg, Ralf: *Meta-learning, model selection, and example selection in machine learning domains with concept drift.* Relatório Técnico, University of Dortmund, 2005. 15

[82] Gama, João e Petr Kosina: *Recurrent concepts in data streams classification.* Knowledge and Information Systems, 40(3):489–507, 2014. 15

[83] van Rijn, Jan N., Geoffrey Holmes, Bernhard Pfahringer e Joaquin Vanschoren: *The online performance estimation framework: heterogeneous ensemble learning for data streams.* Machine Learning, 107(1):149–176, 2018. 15