

Discussão de Implementação do Algoritmo MINAS

Luís Henrique Puhl de Souza

Orientador: Prof. Dr. Hermes Senger

25 de agosto de 2020

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Departamento de Computação

Programa de Pós-Graduação em Ciência da Computação

Introdução

Este documento tem por objetivo apresentar o algoritmo Minas e suas implementações guiando discussões sobre os detalhes e decisões nas implementações.

Recorda-se que o contexto dessa discussão é o mesmo do sistema M-FOG:

- Um sistema para detecção de intrusão em Redes IoT implementando em névoa;
- A hipótese do trabalho é que o algoritmo MINAS pode ser distribuído em nós de nuvem e névoa reduzindo a latência e com pouco comprometimento na qualidade de detecção.
- Fundamentos
 - Métodos Detecção de Novidade;
 - Ambientes de computação Distribuída;
 - Plataformas de processamento distribuído de fluxos.

Algoritmo MINAS

- Modelo de aprendizado *Offline-Online*;
- Transformação dos dados analisados para o espaço \mathbb{R}^d ;
- Modelo de classificação com *Clusters*;
- Função de classificação baseada em distância euclidiana;
- Algoritmo de agrupamento para identificação de novos padrões;
- Classificação de novos padrões entre recorrência, extensão e novidade;

Proposta da Pesquisa

- Implementar a distribuição do algoritmo MINAS em nuvem e névoa conforme arquitetura IDSA-IoT;
- Paralelizar o método de classificação do algoritmo MINAS.

Metodologia

- Plataforma de processamento distribuído;
- Estratégias de implementação da arquitetura IDSA-IoT;
- Experimentação com a distribuição do algoritmo MINAS em ambientes;
- Métricas de qualidade de classificação para validação da implementação;
- Métricas de escalabilidade.

O sistema M-FOG é dividido em 5 módulos subdivididos em 2 grupos.

Módulos principais implementam o algoritmo MINAS

- módulo treinamento (*Training Module*);
- módulo classificador (*Classification Module*);
- módulo detector de novidades (*Novelty Detection Module*).

Módulos auxiliares, utilizados para avaliação

- módulo auxiliar *source* (fonte);
- módulo auxiliar *sink* (sorvedouro, consumidor final).

Proposta

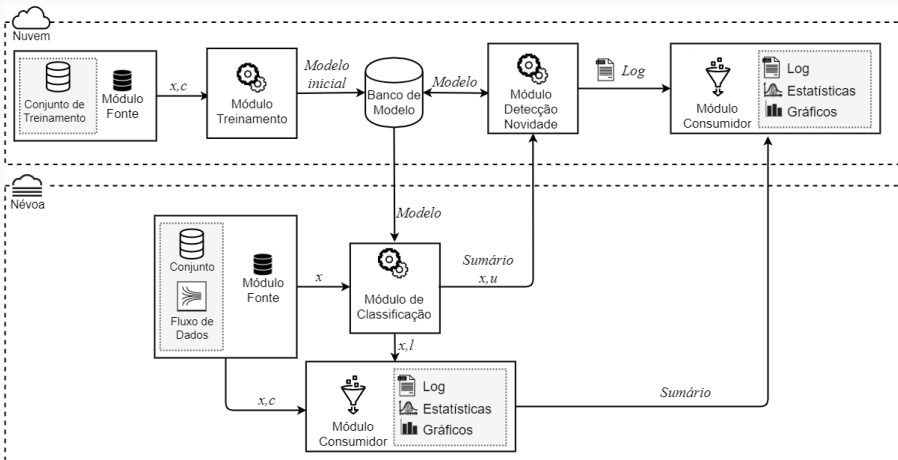


Figura 1: Arquitetura e fluxos de dados do sistema M-FOG.

Métricas e Ambientes

- Métricas de qualidade de classificação:
 - Avaliação do fluxo de saída do classificador;
 - Uso de uma matriz de confusão ou erro;
 - Taxa de desconhecidos;
 - Macro F-score;

$$E_n = \begin{pmatrix} e_{1,1} & e_{1,2} & \cdots & e_{1,J} \\ e_{2,1} & e_{2,2} & \cdots & e_{2,J} \\ \vdots & \vdots & \ddots & \vdots \\ e_{M,1} & e_{M,2} & \cdots & e_{M,J} \end{pmatrix}$$

$$UnkR_n = \frac{1}{M} \sum_{i=1}^M \frac{\#Unk_i}{\#ExC_i}$$

$$Fscore1_n = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Métricas e Ambientes

- Métricas de escalabilidade:
 - Número e tipo de processadores;
 - Uso de memória;
 - Tempo de processamento;
 - Taxa de eventos;
 - Latência entre a produção e classificação.
- Ambientes de teste:
 - Computador Pessoal (para desenvolvimento);
 - Nuvem UFSCar;
 - Nevoa composta de SBC (*Single Board Computer*) ARM 4 núcleos;

Resultados Preliminares

Primeira Implementação com Python e Apache Kafka

- *Python* é acessível e fornece bibliotecas diversas;
- *Apache Kafka* é um sistema de mensagens distribuído;
 - Interface de programação com cliente produtor e consumidor;
 - Mensagens organizadas em tópicos que são distribuídos em partições;
- A hipótese de que a carga seria distribuída entre os consumidores, uma vez que o consumidor pode selecionar uma partição para leitura;
- Em experimento com um produtor, 8 partições e 8 consumidores, observou-se que um consumidor processava a maior parte das mensagens, poucos consumidores recebiam algumas mensagens e a maioria dos consumidores não recebia mensagem alguma.

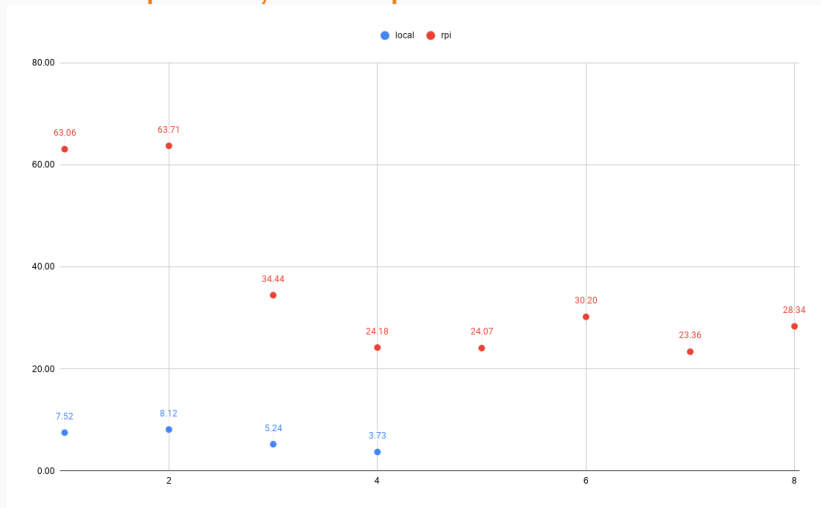
Segunda Implementação com Apache Flink

- Implementação escrita em Scala ou Java;
- Processamento de fluxos *Stateful*;
- Falta de bibliotecas que distribuam algoritmos base como *K-means*;
- Ambiente de execução (*Flink Cluster*) consome mais memória do que disponível no *hardware*;
- Tempo de execução não foi melhor que a implementação original mesmo sem o trecho de detecção de novidades;

Terceira Implementação com OpenMPI

- Implementação escrita em C;
- Versão serial e versão paralela e distribuída com *MPI*;
- Reimplementação de algoritmos base como *K-means*;
- Sistema *M-FOG* em desenvolvimento, atualmente na fase de validação através das métricas de qualidade de classificação.
 - Diferença entre os modelos iniciais gerados pelo algoritmo *K-means*;
 - Diferença na matriz de confusão resultante da avaliação dos fluxos de saída;

Terceira Implementação com OpenMPI



Desafios Atuais

- ~~Diferença entre double e float;~~
- ~~Formato do fluxo de saída;~~
- ~~Tratamento de exemplos com etiqueta *desconhecido* utilizados para atualização do modelo;~~
- ~~Diferença entre incluir ou não a borda do cluster;~~
- ~~Definição de raio;~~
- Distribuição e paralelização para minimização de latência entre novo item no fluxo e sua classificação;
- Detecção de novidades e manutenção de modelo em ambiente distribuído.

Notas de Implementação

Outras abordagens e implementações

- FuzzyND por Da Silva 2018;
- Minas-LC e Minas-BR por Costa 2019;
- Implementação em Java por Douglas
(douglas.m.cavalcanti@gmail.com) em Jul 2 09:37:42 2019
- Implementação em Python por Vitor Sexto Bernardes
(vitorsb@gmail.com) em May 11 23:51:09 2020

Parâmetros

```
class br.ufu.noveltydetection.minas.MinasOg with
    filenameOffline = datasets/training.csv
    filenameOnline = datasets/test.csv

    outputDirectory = out/minas-og//2020-07-20T12-18-21.758/
    algClusteringOff = kmeans
    algClusteringOnl = kmeans

    threshold = 2.0
    flagEvaluationType = 1
    thresholdForgettingPast = 10000
    numMicro = 100
    flagMicroClusters = true

    minExCluster = 20
    validationCriterion = dec

    skipNd = false
```

Parâmetros

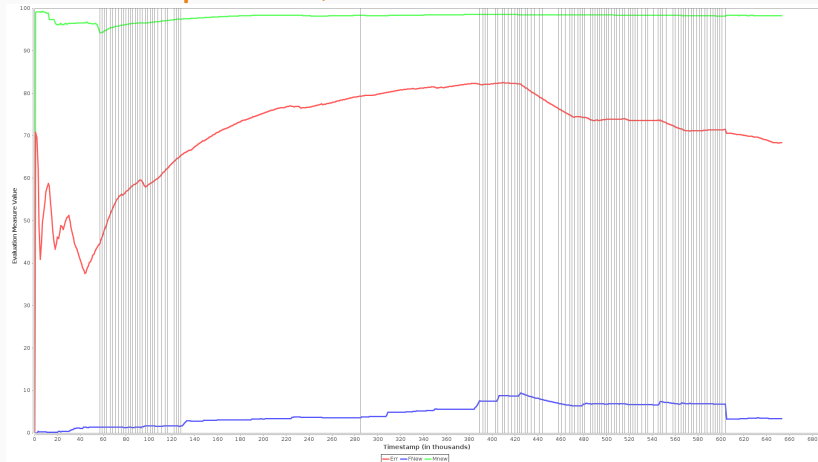
```
params->kParam = 100;  
params->dimension = 22;  
params->noveltyThreshold = 2;  
params->minExCluster = 20;  
params->maxUnkSize = params->kParam * params->minExCluster;  
params->thresholdForgettingPast = 10000;
```

A implementação de referência gera um arquivo (fluxo) de saída e arquivo de matriz de confusão.

Além disso é gerado um gráfico das métricas (Err , F_{New} , M_{new}) calculadas para cada item do fluxo.

Implementação Referência e Avaliação de Referência

Reference Output Stream, Reference Evaluation



Implementação Referência e Avaliação de Referência

Matriz de confusão gerada pela Implementação referência

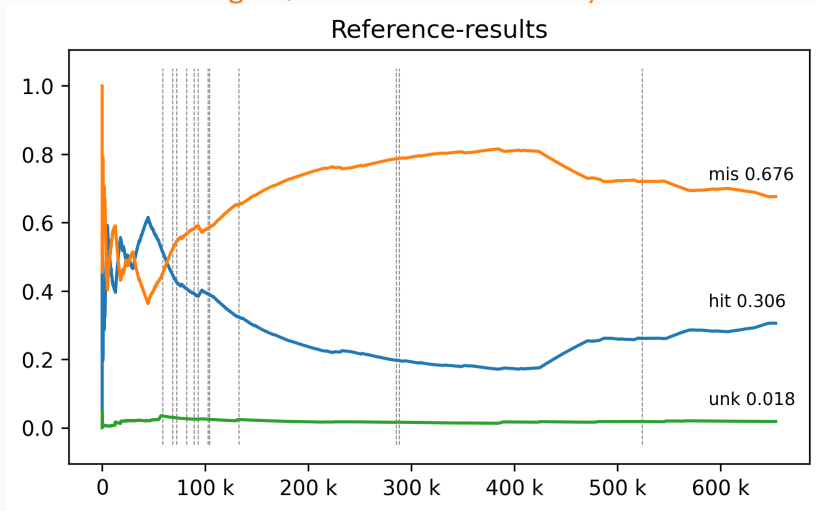
	C N	C A
C N	199263	439530
N 1	0	123
N 2	79	145
N 3	44	368
N 4	0	8
N 5	0	52
N 6	0	165
N 7	229	1
N 8	181	1046
N 9	826	2133
N 10	5088	3467
N 11	289	71
N 12	0	26
Unk	279	44

Aplicando a técnica de associação de etiquetas à classes descrita em ??), tem-se que 208935 itens foram etiquetados corretamente (*true positive*). Portanto a *taxa de acerto* é $208935/653457 = 0.319737948$.

O fluxo de saída da implementação de referência pode ser consumido pelo módulo de avaliação da nova implementação (sistema M-FOG).

Implementação de Referência e Nova Avaliação

Fluxo de saída Original, Grafico da Nova Avaliação



Implementação de Referência e Nova Avaliação

Matriz de confusão e avaliação, Fluxo de saída Original

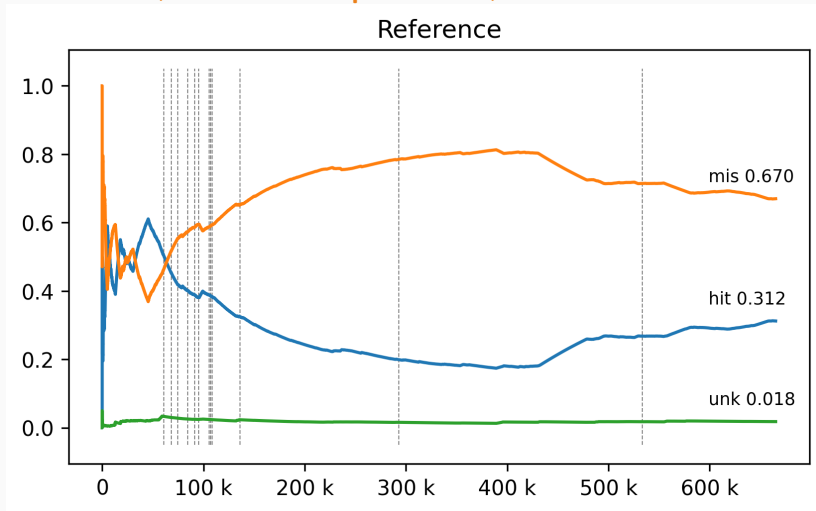
Classes (act) Labels (pred)	A	N
-	3774	8206
1	123	0
10	2489	4066
11	71	289
12	26	0
2	145	79
3	368	44
4	8	0
5	52	0
6	165	0
7	1	229
8	1046	181
9	161	154
N	438750	193030

Métrica	Valor	
Total examples	653457	
Total matches	653457	
Hits	199708	0.30561766
Misses	441769	0.67604907
Unknowns	11980	0.01833326
Unk. reprocessed	0	0.00000000

Além do consumo do fluxo com formato original, foi adicionado à implementação de referência o passo de reprocessamento e saída no novo formato.

Implementação de Referência e Nova Avaliação

New Format, Reference Output Stream, Evaluation



Implementação referência

Matriz de confusão e avaliação, Novo formato Fluxo de saída

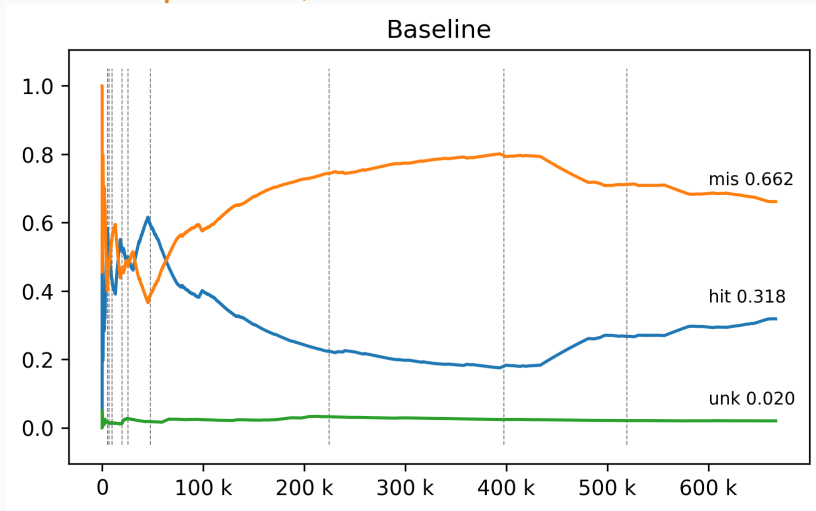
Classes (act) Labels (pred)	A	N
-	3774	8206
1	123	0
10	3520	5130
11	71	289
12	26	0
2	152	82
3	368	44
4	8	0
5	82	1
6	165	0
7	8	396
8	1054	183
9	161	154
N	441395	199715

Métrica	Valor	
Total examples	653457	
Total matches	665107	
Hits	207669	0.31223397
Misses	445458	0.66975389
Unknowns	11980	0.01801214
Unk. reprocessed	11650	0.97245409

Implementação serial, não distribuída, do algoritmo MINAS em C que serve de base (biblioteca) para a implementação sistema M-FOG e para cálculo de speed-up.

Implementação Baseline e Nova Avaliação

Baseline Output Stream, Evaluation



Implementação Baseline e Nova Avaliação

Matriz de confusão e avaliação, Novo formato Fluxo de saída

Classes (act) Labels (pred)	A	N
-	10263	3122
0	1798	48
1	315	0
2	1098	156
3	0	318
4	1510	20
5	2	53
6	1560	2
7	31	0
8	31	3
9	58	3
N	440712	205476

Métrica	Valor	
Total examples	653457	
Total matches	666579	
Hits	212248	0.31841387
Misses	440946	0.66150599
Unknowns	13385	0.02008014
Unk. reprocessed	13122	0.98035114

```
k=100; radiusF = 0.25;  
minExamplesPerCluster = 20;  
noveltyF = 1.4
```

Matriz de confusão e avaliação na Referência versus Avaliação sistema M-FOG

Nota-se que como o novo método de avaliação utiliza o fluxo de saída do algoritmo, as métricas tem valores semelhantes mas não idênticos.

Em especial, a soma de todas as células da matriz de confusão na referência é de 653457 (contagem de itens no fluxo de entrada) e na avaliação atual (sistema M-FOG) é de 665107 (contagem de itens no fluxo de saída).

O mesmo acontece no número de acertos: Na referência 208935 com taxa 0.319737948 e na avaliação atual 207669 com taxa 0.31223397.

Algoritmo MINAS vs Implementação referência

- Definição de raio: desvio padrão das distâncias versus distancia máxima;
- Atualização do micro-cluster limita-se à atualização do atributo T;
- Remoção de exemplos na implementação de referência é feita somente para o algoritmo *CluStream*;
- Inclusão de borda: algoritmo inclui (\leq), referência não inclui ($<$);

Algoritmo MINAS vs Nova Implementação

- Seguiu-se as mesmas divergências anteriores para comparação dos resultados com a implementação referência;
- Inclusão da borda;
- Comportamento do mecanismo de *sleep-model* não está definido, portanto não está ativo;
- Processo de clusterização é limitado ao algoritmo *K-Means*. Algoritmo *CluStream* não está implementado;

Otimizações

Complexidade do cálculo de distância mínima (*Nearest neighbor search – NNS*) de $O(nd)$ para $O(n \log d)$ com integração do cálculo de distância evitando percorrer todas as dimensões se a distância não for mínima;

function	common	fast	ratio
training	6.850 990e1	3.009 816e1	0.439 325 703
noveltyDetection	2.555 810e−1		
minasOnline	1.187 446e1	1.674 746e1	1.410 376 556

Discussões de Implementação

Otimizações

Maximização do número de itens classificados corretamente (*true positive*) por manipulação dos parâmetros:

- *A*: Número mínimo de exemplos por *Cluster* válido;
- *R*: Fator de raio. Multiplica o desvio padrão das distâncias dos elementos do *Cluster* e seu centro;
- *Q*: Fator de novidade. Para distinção entre padrões novidade e extensões;

Observando as métricas:

- *Unk*: Contagem de itens classificados como desconhecidos;
- *Repro*: Contagem de itens classificados como desconhecidos e reprocessados com nova etiqueta;
- *Lbs*: Contagem de etiquetas distintas no fluxo de saída;
- *Hits*: Contagem de itens na matriz de confusão onde a classe atribuída à etiqueta é igual à classe real dos exemplos (*true positive*);
- *Online*: Tempo em segundos (*wall clock*) de execução da fase *online*;

Otimizações: Manipulação de Parâmetros

<i>A</i>	<i>R</i>	<i>Q</i>	<i>Unk.</i>	<i>Repro.</i>	<i>Lbs.</i>	<i>Hits</i>	<i>Online</i>
20	0.25	1.4	12844		15	31.134635%	16.74746
20	0.25	1.4	13385	13122	12	212248 (31.8%)	8.180662
50	0.05	0.20	186433	185853	102	294306 (35%)	30.56771
50	0.05	0.25	186433	185853	74	273008 (32%)	32.22774
25	0.05	0.20	175904	175642	204	279672 (33%)	34.10292
100	0.05	0.20	167496	166759	49	242318 (29%)	29.80112

Obrigado!

