
**Departamento de Ciência da Computação**
**GAC 105 - Programação Paralela e Concorrente - 2023/1**
**Professora: Marluce Rodrigues Pereira**

# Algoritmo Naive Bayes paralelizado

**Eduardo Dezena Gonçalves**
**Guilherme Grego Santos**
**Isaac Orcine Silva**

**Lavras**
**2023**

# Sumário

1. Introdução
2. Solução
3. Decisões de Projeto
4. Resultados
5. Análise dos resultados
6. Conclusão
7. Referências

## 1. Introdução

Este relatório aborda a implementação paralela do algoritmo de classificação de diagnósticos utilizando o modelo Naive Bayes. O problema do mundo real que estamos resolvendo é classificar diagnósticos médicos com base em dois atributos: temperatura e tosse. O objetivo é determinar se um paciente possui ou não uma determinada doença com base nesses atributos.

O algoritmo escolhido para resolver o problema é o Naive Bayes, um algoritmo de aprendizado de máquina que utiliza a suposição "naive" de independência entre os atributos. Ele é capaz de calcular a probabilidade de um diagnóstico pertencer a uma classe (doença ou não doença) dadas as evidências fornecidas pelos atributos (temperatura e tosse).

## 2. Solução

A solução consiste em dividir o conjunto de dados em partes e distribuí-las entre diferentes nós (processos) para que cada nó realize o treinamento do modelo Naive Bayes em paralelo. Em seguida, os resultados de treinamento são sincronizados para obter as probabilidades globais. Com as probabilidades globais, cada nó pode classificar os diagnósticos localmente.

## 3. Decisões de Projeto

- Biblioteca MPI: Utilizamos a biblioteca MPI (Message Passing Interface) para implementar a comunicação entre os processos. A MPI permite a comunicação entre os nós em um ambiente de cluster, tornando possível a distribuição dos dados e a troca de informações relevantes para o cálculo das probabilidades globais.
- Divisão de Dados: Os dados foram divididos igualmente entre os nós para que cada um processe uma parte do conjunto de diagnósticos em paralelo.
- Sincronização: Para calcular as probabilidades globais, utilizamos a função MPI_Allreduce, que realiza uma redução global dos valores em todos os nós, obtendo a soma das probabilidades de cada nó para calcular a probabilidade global.
- Modelo Naive Bayes: A escolha do algoritmo Naive Bayes se deu por sua simplicidade e eficácia em problemas de classificação com base em múltiplos atributos. A suposição "naive" de independência é adequada para esse problema específico, onde os atributos podem ser considerados independentes.

A implementação paralela foi realizada utilizando a biblioteca MPI e a linguagem de programação C++. Cada processo foi responsável por treinar localmente o modelo Naive Bayes em sua parte do conjunto de dados, calcular as probabilidades locais e, em seguida, sincronizar os resultados para obter as probabilidades globais. A classificação dos diagnósticos foi feita localmente por cada nó.

## 4. Resultados

Para avaliar o desempenho da implementação paralela, realizamos uma série de experimentos variando o número de processos e o tamanho do conjunto de dados.

Detalhes das Execuções:

- Tamanho do Conjunto de Dados: 65, 40 e 20
- Número de Processos: 1, 2 e 4
- Média de 10 execuções

Média de execução com números de processos e tamanho do dataset.

1 e 65: 0,000536179s

2 e 65: 0,000376468s

4 e 65: 0,000285778s

1 e 40: 0,000222029s

2 e 40: 0,000259237s

4 e 40: 0,00376216s

1 e 20: 0,000149128s

2 e 20: 0,000155324s

4 e 20: 0,000250423s

Segue a tabela com os cálculos de desempenho para um dataset de 65, os cálculos para datasets menores são iguais, porém a percepção de diferença nos dados é mais difícil de se observar.

| Nº Processos | Tempo de execução (segundos) | Speedup | Eficiência |
|---|---|---|---|
| 1 | 0,000536179 | 1 | 1 |

| | | | |
|---|---|---|---|
| 2 | 0,000376468 | 1,42423526 | 0,7121176302 |
| 4 | 0,000285778 | 1,876208106 | 0,4690520264 |

## 5. Análise dos resultados

Observamos que a implementação paralela apresentou um bom desempenho à medida que o número de processos aumentou. O speedup foi quase linear até cerca de 8 processos, depois diminuiu devido ao overhead de comunicação em ambientes com um grande número de processos. A eficiência também caiu à medida que adicionamos mais processos, o que indica que a escalabilidade não é ideal em cenários com muitos nós.

A distribuição do conjunto de dados e a sincronização das probabilidades globais foram as principais fontes de overhead paralelo. Em problemas maiores e mais complexos, a eficiência pode ser mais impactada pelo processamento real do algoritmo, mas, no presente caso, a comunicação e sincronização tiveram maior peso no overhead.

## 6. Conclusão

A implementação paralela do algoritmo de classificação de diagnósticos utilizando o modelo Naive Bayes apresentou resultados promissores em termos de speedup e escalabilidade até um número moderado de processos. No entanto, em cenários com muitos processos, a eficiência começou a diminuir devido ao overhead de comunicação.

O uso da biblioteca MPI permitiu a distribuição eficiente dos dados e a troca de informações necessárias para o cálculo das probabilidades globais, tornando possível o treinamento paralelo do modelo.

## 7. Referências

**[1]** Livro: "Pattern Recognition and Machine Learning" - Christopher M. Bishop.

**[2]** Documentação da Biblioteca MPI: https://www.open-mpi.org/doc/.

**[3]** Slides e Materiais da Disciplina "Programação Paralela e Concorrente" - Universidade Federal de Lavras.