Relatório de Projeto 2: Modelos Markovianos Ocultos

Tema: Sistema de Apoio ao diagnóstico de falhas em robôs aspirador.

Disciplina: Inteligência Artificial (FGA0221)

Professor: Fabiano Araujo Soares, Dr

Aluno: Marcos Antonio Teles de Castilhos

1. Introdução e Objetivo

Enquanto o Projeto 1 focou na decisão de compra (um problema estático), a operação real de um robô aspirador é um processo dinâmico. Frequentemente, o usuário se depara com logs de erros ou comportamentos estranhos onde a causa raiz não é clara.

O objetivo deste projeto é desenvolver um sistema de diagnóstico temporal utilizando Modelos Markovianos Ocultos (HMM). O desafio central é distinguir estados operacionais críticos (como "Preso") de estados normais (como "Base"), mesmo quando eles geram leituras de sensores idênticas (ex: "EmEspera"), utilizando a história de eventos para desambiguação

Este tema foi escolhido pela dificuldade real de interpretar sensores individuais fora de contexto. Uma leitura de "colisão", por exemplo, pode ser normal durante a limpeza ou indicar que o robô está preso, dependendo do histórico recente.

2. Fundamentação Teórica: Hidden Markov Models

Os Modelos Markovianos Ocultos (HMM) são modelos probabilísticos usados para descrever sistemas que transitam entre estados (ocultos) ao longo do tempo, gerando observações visíveis a cada passo.

O modelo baseia-se em duas suposições fundamentais:

1. Suposição de Markov: O estado futuro depende apenas do estado atual, não de todo o histórico.

2. Independência da Observação: A observação atual depende apenas do estado atual.

Para resolver este problema, modelamos o sistema como um HMM, composto por:

1. Estados Ocultos (Q): A realidade não observável (Limpando, Preso, Base).

2. Observações (O): A leitura ruidosa dos sensores (Normal, Colisão, EmEspera).

3. Dinâmica (Matrizes A e B): As regras probabilísticas de como o robô se move e como os sensores reagem.

Utilizamos o Algoritmo de Viterbi, um método de programação dinâmica que computa o "caminho mais provável" (argmax P(Q|O)) através da treliça de estados, garantindo que a explicação mais coerente globalmente seja escolhida, e não apenas a mais provável localmente.

3. Metodologia (O Modelo HMM)

A modelagem focou na captura da física e lógica do robô através das matrizes de probabilidade:

3.1 Matriz de Transição (A) - A Lógica Temporal

Definimos probabilidades que respeitam a continuidade física.

- Inércia: Um robô 'Limpando' tende a continuar 'Limpando' (80%).
- Impossibilidades Físicas: Definimos probabilidade zero (0.0) para transições diretas de 'Preso' para 'Base'.

Isso ensina ao modelo que o robô não pode se teletransportar para o carregador se estiver enroscado em um fio.

3.2 Matriz de Emissão (B) - O Modelo do Sensor

- Ambiguidade Modelada: O estado 'Preso' gera a observação 'EmEspera' com 40% de chance, e o estado 'Base' gera a mesma observação com 90% de chance. É papel do algoritmo distinguir qual dos dois está ocorrendo.

4. Análise de Resultados e Comparação

O modelo foi submetido a um cenário de teste projetado para induzir erro em sistemas simples.

Sequência de Log: ['EmEspera', 'Normal', 'Normal', 'Colisao', 'Colisao', 'EmEspera']

4.1 Comparação: HMM (Viterbi) vs. Diagnóstico Ingênuo

Implementamos um "Diagnóstico Ingênuo" (Baseline) que olha apenas para o sensor atual, ignorando o passado.

- Tempo 0 ('EmEspera'): Ambos acertam ('Base').

- Tempo 3 ('Colisao'): O Ingênuo alerta 'Preso' imediatamente. O HMM, usando o contexto, entende quecolisões eventuais são normais durante a limpeza e mantém o estado 'Limpando'.

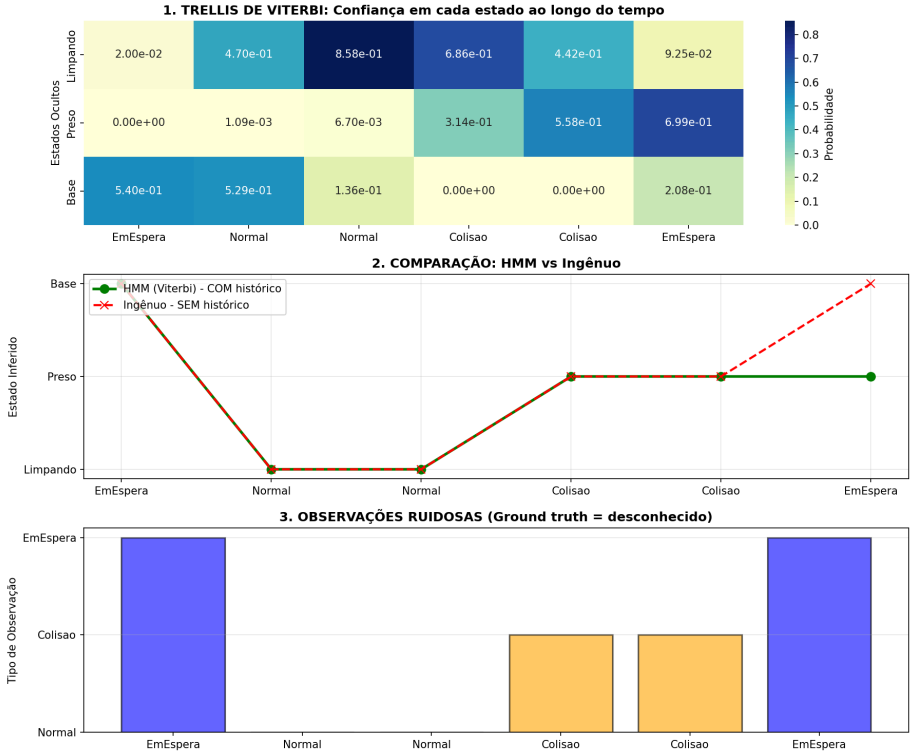
- Tempo 5 ('EmEspera'): O Ponto Crítico.

> O Ingênuo vê 'EmEspera' e conclui erroneamente que o robô voltou para a 'Base'.

> O HMM analisa o histórico de colisões recentes e a impossibilidade de transição Preso->Base. Ele conclui corretamente que o robô desistiu e está 'Preso'.

4.2 Análise Visual

O gráfico gerado ('analise_hmm_completa.png') apresenta a Trellis de Viterbi (mapa de calor), evidenciando como a probabilidade flui e se concentra no caminho correto, descartando caminhos fisicamente impossíveis apesar das leituras confusas dos sensores.



5. Guia de Instalação e Execução

Para reproduzir os resultados deste projeto, siga as instruções abaixo.

Pré-requisitos

- Python 3.12 ou superior.
- Bibliotecas: numpy, matplotlib, seaborn, pandas.

Passo 1: Configuração do Ambiente

Recomenda-se o uso de um ambiente virtual para isolar as dependências.

No Windows (PowerShell):

PowerShell

1. Crie o ambiente virtual

python -m venv venv

2. Ative o ambiente

.\venv\Scripts\activate

No Linux/Mac:

Bash

python3 -m venv venv

source venv/bin/activate

Passo 2: Execução do Código

Com o ambiente ativado, execute o script principal.

Bash

python hmm-v2.py

Passo 3: Saída

- Console: Tabela comparativa detalhada e explicações passo a passo da lógica.

- Arquivo: 'analise_hmm_completa.png' contendo a visualização da Trellis e a comparação das trajetórias inferidas

6. Conclusão

Este projeto comprovou a superioridade dos modelos probabilísticos temporais sobre abordagens estáticas para diagnóstico. O HMM foi capaz de "contar uma história" coerente sobre o comportamento do robô, transformando dados brutos em informação acionável. A capacidade de desambiguar o estado final 'Preso' do estado 'Base' apenas pelo contexto histórico demonstra o valor prático dessa técnica para aumentar a confiabilidade de sistemas autônomos.