Projeto 3 - Programação Dinâmica

SCC-5900 - Projeto de Algoritmos

Aluno: Afonso Matheus Sousa Lima

Nº USP: 11357812

1 Explicação da Implementação

O *Dynamic Time Warping* (DTW) é um algoritmo baseado em programação dinâmica utilizado, principalmente, para comparar duas séries temporais, conseguindo identificar as distorções temporais entre elas. Em sua essência, utilizando uma abordagem *botton-up*, o DTW utiliza uma matriz de distâncias com dimensões m, n das duas séries temporais que serão comparadas. Essa matriz será preenchida com as distâncias entre os pontos das séries, sendo que ao atingir a posição $[m, n]$ da matriz, ela terá terminado de computar o valor total da distância entre as duas séries temporais.

Um exemplo ilustrativo é apresentado a seguir. Suponha duas séries temporais A e B, onde $A = 1, 2, 1$ e $B = 3, 4, 4, 3$ (Figura 1).

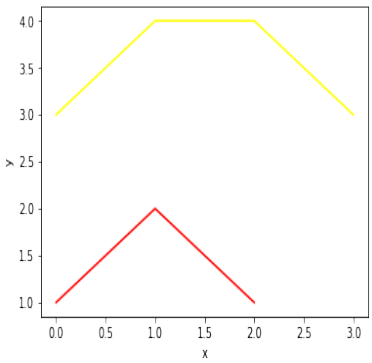

Figura 1: Séries temporais A e B

Utilizando o DTW, será criado uma matriz com as dimensões de A e B, mais uma linha e coluna para valores auxiliares que facilitarão a execução do algoritmo. Então, para cada valor $i \in A$ será calculada a distância com cada valor $j \in B$ utilizando a função de distância proposta pelo projeto $(A[i]-B[j)^2$. Essa distância será somada com o valor mínimo entre as posições abaixo ($[i, j-1]$), à esquerda ($[i-1, j]$) e na diagonal

inferior ($[i-1, j-1]$) da posição que está sendo computada em um determinado momento. Note que quando a matriz está vazia e vai computar a primeira posição, os valores abaixo e à esquerda são preenchidas com um valor muito alto, e a posição na diagonal é preenchida com 0. Isso fará com que a primeira posição seja somente a distância entre os primeiros valores das séries. Esses valores muito altos também contribuem para o calculo das posições referentes a primeira linha e primeira coluna, que não teriam valores abaixo/diagonal e esquerda/diagonal, respectivamente, para computar.

Com isso, a matriz será preenchida com as soma das distâncias possíveis entre os pontos das duas séries. Finalmente, na posição $[m, n]$ da matriz, teremos o valor da similaridade entre as duas séries, sendo que quanto menor esse valor, mais semelhantes elas são. A matriz preenchida pode ser vista na Figura 3. As posições em negrito representam os valores que foram sendo somados para chegar ao valor de similaridade final. Com essas posições também podemos traçar as retas que relacionam a similaridade dos pontos, que pode ser vista na Figura 2.

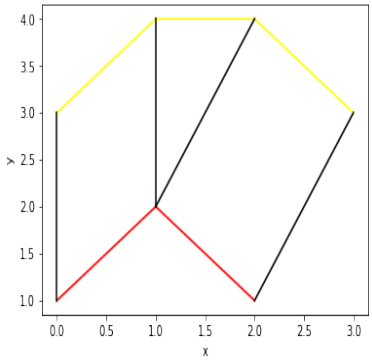

Figura 2: Semelhanças entre os pontos de A e B

1	∞	9	14	22	**16**
2	∞	5	**8**	**12**	13
1	∞	**4**	13	22	26
	0	∞	∞	∞	∞
		3	**4**	**4**	**3**

Figura 3: Matriz de distâncias entre A e B

Utilizando a semelhança entre quaisquer duas séries temporais obtida pelo DTW, pode-se realizar a classificação proposta pelo projeto. Foi implementado um algoritmo *K-Nearest Neighbors* (KNN) onde K é igual a 1. Para toda série temporal pertencente a base de teste será verificado, utilizando o DTW, qual a série temporal pertencente á base treino mais semelhante e, com isso, a série de basa de treino será rotulada com a mesma classe dessa série mais semelhante. Ao final, será feito a comparação entre os rótulos preditos pelo algoritmo e os rótulos reais, obtendo assim a acurácia do classificador.

2 Executando o Algoritmo

O algoritmo foi desenvolvido utilizando Python3. Ele pode ser executado utilizando o comando:

```
python3 ts_classification
```

Será apresentado o progresso de classificação das séries temporais pertencentes a base de teste, mostrando qual a ultima série classificada, de um total de 960 séries. Também é apresentado o tempo de classificação (segundos) para cada série temporal conforme o algoritmo for sendo executado. Ao final, é apresentado o tempo total de execução e a acurácia do algoritmo de classificação.

3 Resultados

Para classificar o todas as 960 séries temporais presentes na base de teste, foi necessário **aproximadamente 28 minutos** de execução. Esse tempo de execução relativamente alto pode ser atribuído tanto à complexidade do algoritmo DTW ($O(m * n)$), onde m e n são os tamanhos das séries temporais, como à complexidade do KNN, onde cada série temporal na base de treinamento será comparada com todas as séries da base de treino, ou seja, sua complexidade será $O(M * N)$ onde M e N são as quantidades de séries na base de treinamento e teste.

O classificador obteve **aproximadamente 85%** de acurácia para a classificação da base de teste. Embora seja um bom valor de acurácia, dentro do âmbito de jogos, é necessário buscar exatidões cada vez maiores para detectar o movimento do jogar, pois qualquer erro de reconhecimento pode gerar o descontentamento do usuário.

Pode-se observar também que cada série pode levar, inicialmente, entre 0.6-1.2 segundos (utilizando a máquina do autor deste relatório) para ser classificada. Com isso, ainda dentro do âmbito de jogos, esse tempo de detecção de um movimento pode ser considerado lento dependendo da dinâmica do jogo em questão. Para jogos que não exigem respostas rápidas dentro da sua jogabilidade, o classificador implementado ainda é interessante, porém caso contrário, o tempo de resposta também pode frustra o jogador.