Universidade de São Paulo

SCC0602 - Algoritmos e Estruturas de Dados I

Projeto - Programação Dinâmica

Relatório Técnico

Autores:
Bruno Ribeiro Helou Nº USP: 10276852
Leandro Giusti Mugnaini Nº USP: 10260351

2018

Sumário

1 Introdução

Algoritmos de classificação são amplamente utilizados nas áreas de processamento de imagens, processamento de linguagem natural, ciência de dados, inteligência artificial, robótica, entre outras. Conseguir classificar dados coletados comparando-os com dados já existentes é uma ferramenta extremamente útil que poupa tempo e recursos computacionais.

Nesse trabalho, o objetivo é classificar séries temporais dentre um conjunto de classes de movimentos realizados com um controle do videogame Wii. Para isso, implementou-se na linguagem *Python* um algoritmo DTW ("*Dynamic Time Warping*"), que utiliza programação dinâmica para estimar a distância entre duas séries temporais. Logo em seguida, aplicou-se o KNN ("*k-Nearest-Neighbors*") que tem o objetivo de classificar as séries temporais de acordo com os K-vizinhos mais próximos.

2 Referencial Teórico

Nessa seção será apresentada uma breve descrição do funcionamento dos algoritmos de DTW e KNN.

2.1 *Dynamic Time Warping* (DTW)

O DTW é um algoritmo que possui o objetivo de comparar duas séries temporais e realizar um alinhamento dessas séries. Esse algoritmo pode ser utilizado para comparar uma série de diferentes dados que obedeçam uma ordem temporal, como vídeos, áudio, e imagens. No caso desse projeto, os dados são de um acelerômetro de um controle de Nintendo Wii.

O algoritmo de DTW pode utilizar diversas métricas de distância, para medir a similaridade entre as séries temporais, como a Euclidiana. Com as características extras das séries, é possível obter uma maior precisão na hora de se classificar os dados, comparando-os com dados pré-existentes (conjunto de treinamento), como é o caso desse projeto, que utiliza o DTW para extrair as características e alinhar as séries temporais, e o KNN (que será explicado logo abaixo), para classificar as séries de acordo com esses dados.

2.2 *(KNN- k-Nearest-Neighbors)*

O KNN é um algoritmo que possui a finalidade de classificar séries, de acordo com o grau de proximidade entre elas, comparando os k vizinhos mais próximos. Nesse projeto, por exemplo, 240 séries temporais servem como a base de dados fixa, isto é, a base de dados treinada dos movimentos, outras 960 séries temporais são utilizadas como teste. Se for escolhido um valor de k=1, o algoritmo KNN comparará a distância de toda a série do teste, com todas as séries do treino, encontrando a menor distância entre elas.Se k=5, o

algoritmo realizará o mesmo procedimento anterior, só que agora escolherá a série com as 5 menores distâncias.

Escolhendo as séries mais próximas, é possível classificar as séries do teste de acordo com o rótulo da série do treino. Ao fim, pode-se realizar uma análise da precisão do algoritmo, calculando o número de acertos do algoritmo sobre o número total de classificações, para diferentes valores de k.

3 Análise do Algoritmo

Para explicar o funcionamento do código, analisaremos o que cada função e estrutura de repetição faz individualmente.

Para começar temos a função "funcaocriterio". Esta função recebe uma lista(matrizauxiliar) de tamanho 13 (indo da posição 0 a 12), onde em cada posição dessa lista está o número de vezes que um rótulo com valor da posição aparece dentre os k-menores. Por exemplo: temos a seguinte lista para um valor de k=16 [0,3,4,3,1,5,0,0,0,0,0,0,0], o código entende que há 0 rótulos com valor '0', 3 rótulos com valor '1',4 rótulos com valor '2' e assim por diante, dentre os 16 menores rótulos presentes na 'matrizurna' ordenada. Por fim ela escolhe qual desses valores do número de rótulos é maior e passa o valor da posição para ser comparado com o rótulo da lista teste que está sendo utilizada. Em caso de empate, o código gera novamente a matriz 'matrix', aumentando K em 1 até que haja desempate dos 'n' maiores na 'matrizauxiliar'.

Em seguida temos a função 'makematrix'. Ela é responsável por criar a matriz 'matrix', cujos valores serão os k-menores rótulos presentes na 'matrizurna'.

Desses valores presentes em matrix, temos então a função 'geramatrizauxiliar', responsável por criar a lista utilizada na 'funcaocriteiro' explicada acima.

Seguindo o código, temos um 'input' de um valor que será o responsável por receber o valor de 'k' desejado.Logo a seguir são declaradas as listas que serão utilizadas para calcular o DTW e para tomarmos os rótulos, abertas utilizando a biblitoeca 'csv'. Também nessa parte do código iniciamos os contadores 'correto' e 'errado', além de retirar os rótulos das listas que serão usadas para o DTW.

Já na linha 78, temos um *while* que será responsável por mudar a linha da lista teste usada para calcular o DTW, que só mudará após todo o processo ocorrer para a linha anterior.

No próximo *while*, teremos a mudança de linha da lista treino. Portanto para cada linha da lista teste calcularemos o DTW com todas as linhas da lista treino.

O cálculo do DTW começa a seguir com os dois próximos *while* onde estará definida a operação para o cálculo a ser realizado. Essa parte foi realizada imaginando que uma lista forma a coluna e a outra as linhas, assim por se tratar de Python, armazenamos os valores calculados em uma lista de listas(matriz2), onde cada lista representa uma linha da matriz imaginária. Nessa matriz imaginária tomamos o seu ultimo valor, bem

como o rótulo da lista treino utilizada para seu cálculo e armazenamos numa nova lista, a 'matrizurna'. Após realizar o cálculo de todos os DTW's e armazená-los junto com os rótulos da 'matrizurna', a mesma é ordenada do menor para o maior. Assim criamos a matriz 'matrix' já explicada anteriormente e geramos a matriz auxiliar.

Na próxima linha chamamos a função 'funcaocriteiro', que retornará o valor a ser comparado com o rótulo da lista teste. Caso os valores sejam iguais, correto recebe o acréscimo de 1, caso contrário, errado recebe esse acréscimo. Assim termina-se o cálculo para o primeiro teste, e inicia-se a próxima linha da lista teste até que ela acabe. Então dividimos o numero de corretos pelo numero total de linhas teste(960), assim temos a porcentagem de acertos para dado K escolhido.

Podemos concluir também analisando o código, que a complexidade total do programa será da ordem de O(n^4), por conta das estruturas de repetição do programa, que em seus piores casos terão a ordem de n (ao todo, são 4 estruturas de repetição, culminando em uma complexidade de O(n^4). As outras partes do código por terem uma complexidade menor são desconsideradas no cálculo da complexidade.

4 Análise dos Resultados

Os números de acertos e erros estão colocados na tabela abaixo:

Valor de K	Acertos	Erros	Porcentagem de acerto (Em %)
1	812	148	84,58
3	783	177	81,56
5	747	213	77,81
10	699	261	72,81
20	615	345	64,06

Pode-se perceber que quanto mais se aumentava o K, maior ficava a taxa de erro, isso se deve pois a comparação entre a série temporal do movimento teste com o rótulo 1 e treino com rótulo 1, tende a ser mais parecida (com um DTW menor), assim quando comparamos valores pequenos de K pegamos as séries que mais se parecem e contamos a quantidade de rótulos parecidos. Ao pegar k com valor grande, as séries com rótulos iguais mais parecidos e com rótulos diferentes são contados. Assim, mesmo que os menores valores do DTW representem um rótulo, ao tomarmos esse k, estamos dando mesmo peso para as séries, fazendo com que o rótulo errado seja atribuído ao teste, contabilizando um erro.

Os tempos medidos nas etapas do código foram colocados na tabela abaixo. Para os valores de tempo de ordenação, do KNN e do DTW foi calculado um tempo médio para 4 valores. O tempo total também foi calculado.

Valor de K	Tempo Médio (Em seg.)			Tempo Total (Em min.)
	Ordenação	KNN	DTW	
1	6.7317e-05	7.9693e-05	0.0029	16,21
3	7.3958e-05	8.8750e-05	0.0030	16,48
5	6.7015e-05	8.3014e-05	0.0031	16,63
10	6.5506e-05	8.3316e-05	0.0031	16,52
20	0.0001	0.0002	0.0031	16,99

Pode-se concluir que o tempo de ordenação não depende do K escolhido, pois as listas com todos os valores já estarão ordenadas, e só depois que o valor de K será utilizado. Como a execução do DTW não depende do valor de K, os valores também não variaram de forma significativa para os diferentes valores de K. Já o KNN é uma execução linear, o que faz com que quando se aumente o valor de K, o valor do tempo não se altere significativamente.

Os valores do tempo total variaram levemente, por conta de processos que estavam sendo executados em segundo plano pelo sistema operacional.

5 Conclusão

Podemos concluir que o algoritmo KNN se mostrou efetivo para um valor pequeno de K. Para valores grandes a serem comparados, outros algoritmos de classificação mais complexos talvez executem a tarefa de modo mais eficiente.