UNIVERSIDADE DE SÃO PAULO – USP
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE SISTEMAS DE COMPUTAÇÃO

RELATÓRIO
Projeto 2 - Método DTW

Aluno: Adams Vietro Codignotto da Silva - 6791943

São Carlos
2013

1 Introdução

Neste projeto, foi implementado o método DTW (Dynamic Time Warping) para avaliar a classificação de séries temporais. O código foi desenvolvido em C, utilizando o Linux Xubuntu 13.04 64bits. O arquivo compactado possui um Makefile, com os comandos *Makeall2* para compilar a primeira vez, *Makeall* para consecutivas vezes (basicamente, este contém um comando Clean antes da compilação) e Make run para iniciar o programa.

2 Modelagem do problema

Primeiramente foi necessário ler e identificar todas as séries temporais dos arquivos, sendo estes *teste.txt* e *treino.txt*. Ambos os arquivos foram lidos e carregados na memória, com o intuito de deixar o programa muito mais rápido, uma vez que a quantidade de dados é grande, e a leitura repetida dos dois arquivos deixaria a execução muito mais lenta.

O método DTW utilizado foi baseado no método já existente na Wikipedia, e adaptado para o problema em questão.

Este método consiste em, dadas duas séries temporais r e t, achar o melhor mapeamento com a menor distância possível entre elas, utilizando de Programação Dinâmica, e a distância euclidiana entre os pontos. O objetivo do DTW é encontrar um caminho tal que a distancia do mapeamento é minimizada, respeitando as fronteiras finais e iniciais das séries, e as restrições de cada ponto, uma vez que as únicas três possibilidades são: dado um ponto (i,j), podemos apenas avaliar os pontos (i-1,j),(i,j-1) e (i-1,j-1). Isso garante que todos os pontos da série r terão pelo menos um elemento correspondente na série t.

Podemos resumir o algoritmo do DTW em:

1. Definir uma matriz de tamanho MxN, dados M e N tamanhos de vetores u e v.
2. Definir a distância D(i,j) como a distância DTW entre t(1,i) e r(1,j), inicialmente em (1,1) e terminando em (m,n).
3. Calcular D(i,j) como sendo $|t(i) - r(j)|^2 + minDTW(i-1,j), DTW(i-1,j-1), DTW(i,j-1)$ para todos os pontos (i,j) dos intervalos.
4. O resultado será $D(m,n)$

O DTW possui complexidade $O(m * n)$. Para este trabalho especificamente, o programa obteve ordem de $O(n^2)$, pois para cada linha do arquivo de teste, precisamos percorrer todas as linhas do arquivo de treino.

Obs.: o arquivo *rtulos.txt* não foi utilizado em nenhum momento, pois não houve necessidade de associar as classes ao que elas representam.

3 Experimentos e Resultados

O código foi implementado primeiramente fazendo a leitura dos arquivos durante todas as iterações, com o intuito de diminuir o uso de memória. Porém o ganho de tempo para execução compensou a quantidade de memória RAM utilizada para armazenar os arquivos. Deste modo, o programa levou aproximadamente 30 segundos para ser executado. Se mantivessemos a leitura de arquivos, este levaria aproximadamente 4 minutos.

Após isso, procuramos a série no arquivo *treino.txt* que mais se aproximava da série no arquivo *teste.txt*, e a reclassificamos, salvando novamente a classificação na RAM. Após todas as reclassficações, comparamos as novas classificações com as feitas previamente pelo método 1-vizinho mais próximo.

O método DTW se mostrou bastante eficaz na classificação das séries, com uma rápida execução da sua versão iterativa, sendo superior ao método préviamente utilizado e até mesmo mostrando que as classificações anteriores não estavam 100% corretas. Utilizando o novo método de classificação mostramos que, das 960 classificações feitas, haviam 808 classificações corretas, assim o método anterior de fato é pior, com uma quantidade de acerto de 84,17%.

4 Conclusões

O método DTW apresentou ótimos resultados. Porém, ele pode ser melhorado, pois apresenta um custo computacional elevado para séries muito grandes, assim como a maioria dos métodos de programação dinâmica (por exemplo, o algoritmo Djikstra e o algoritmo Djikstra A*).

Como nesse projeto foram usadas relativamente poucas e pequenas séries, uma opção foi salvar todos os dados na memória. Num caso em que tal abordagem não seja possível, um bom modo de otimizar o DTW seria fazendo o relaxamento de suas fronteiras, suavizando picos e altas taxas de variações, assim diminuindo a quantidade de pontos a serem comparados e cosequentemente, minimizando o custo computacional do método.