Universidade Federal do Paraná
Curso de pós-graduação em Informática
Disciplina: Inteligência Artificial
Professor: Fabiano Silva
Aluna: Angelita Rettore de Araujo Zanella

Trabalho 1
Classificação de dados de sensores

Objetivo

Este trabalho tem como objetivo identificar anomalias em dados de sensores a fim de isolar tais dados para que não sejam utilizados durante a tomada de decisões por sistemas inteligentes.

Descrição do programa

O classificador foi implementado em linguagem C e utilizou o algoritmo DBSCAN para isolar *outliers*. O código original do DBSCAN foi modificado para acomodar um ajuste de heurística, realizado por um algoritmo genético que otimiza o valor relacionado à vizinhança. Originalmente, o DBSCAN recebe como parâmetros os valores *minps* e *eps*. O primeiro define a quantidade mínima de vizinhos que um dado deve possuir para ser considerado '*core'*. O segundo define qual é a distância euclidiana máxima entre dois dados para que sejam considerados vizinhos. Como os dados dos sensores estão relacionados às temperaturas lidas em uma estufa agrícola, utilizou-se as temperaturas lidas por sensores para o cálculo da distância.

Uma vez que as temperaturas variam ao longo do ano, o DBSCAN cria vários clusters para agrupar temperaturas das diferentes estações do ano. Visando melhorar o agrupamento dos dados e ajustar o valor de *eps* às variações climáticas, foi utilizado um algoritmo genético que recalcula o perímetro da vizinhança dentro de um limite pré-estabelecido, denominado *neighbor*. O algoritmo DBSCAN continua funcionando como o original, ou seja, a definição de nós como core ou *outlier* não foi modificado. Porém, quando são encontrados nós a uma distância entre esp e eps+neighbor, os dados desse nó são armazenados em um vetor para ajuste do valor de eps. Quando existirem pelo menos 10 (dez) novas distâncias, é iniciado o processo de ajuste, seguindo os passos a seguir:

1. Os dez valores maiores que eps e menores que eps+neighbor são recuperados da memória;
2. Os dois valores mais altos são descartados (porque não queremos que o novo valor de eps seja muito superior ao original);
3. Os dois valores mais baixos são 'clonados' e passam a compor os cromossomos da nova população;
4. Um desses novos cromossomos sofre mutação (soma-se 1 ao valor binário do cromossomo);
5. Os valores originais, exceto os dois mais altos, são selecionados como cromossomos para operação de crossover. Nesta operação, é realizada a troca dos 4 bits mais à direita entre os dois cromossomos.

Então a nova população é composta por dois cromossomos relacionados aos dados originais, sendo que um sofreu mutação, e oito cromossomos que passaram pelo processo de

crossover, totalizando dez novos indivíduos. Finalmente, os valores obtidos são somados e tem sua média calculada. O valor médio será utilizado como novo valor de eps.

Compilação

Para compilar o programa, basta utilizar o comando make.

Executando

Para executar utilize:

```
./anomalydetection <database> <minps> <eps> <neighbor>
```

Database: base de dados contendo as leituras (valores numéricos em formato decimal (int ou float))

Minps: número mínimo de vizinhos para um dado ser considerado core

Eps: distância máxima para um dado ser considerado vizinho

Neighbor: limites da vizinhança de eps.

Exemplo:

```
./anomalydetection database/3303.csv 10 5 1
```

O resultado será salvo nos arquivos **ad-results.csv** e **ad-results.txt**. O primeiro dispõe os dados no formato [dado];[cluster] e o segundo informa o cluster e todos os dados que compõem o cluster informado.

Obs.: o diretório database contém arquivos da base de dados utilizados para testes desse algoritmo. O arquivo 3303.csv está no formado adequado para este trabalho. O algoritmo não faz validação de entrada e o formato do arquivo database não é verificado. Sugere-se que os dados do arquivo de entrada estejam dispostos individualmente em cada linha (um em cada linha).