Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação
SCC0201 – Introdução à Ciência da Computação II

Trabalho 4: K-Nearest Neighbors (KNN)

Professor: Dr. Rodrigo Fernandes de Mello (mello@icmc.usp.br)
Dr. Moacir Antonelli Ponti (moacir@icmc.usp.br)
Estagiário PAE: Felipe Simões Lage Gomes Duarte (fgduarte@icmc.usp.br)
Gabriel de Barros Paranhos da Costa (gbpcosta@icmc.usp.br)
Tiago Santana de Nazaré (tiagosn@usp.br)

1 Objetivo

Você deverá implementar o algoritmo de classificação K-Nearest Neighbors (KNN) tomando como base os trabalhos anterirores.

2 K-Nearest Neighbors (KNN)

O algoritmo de classificação k-vizinhos mais próximos (do inglês *k-Nearest Neighbors* – KNN) é um dos mais simples e, ainda sim, mais utilizados na área de Aprendizado de Máquina. Este algoritmo utiliza uma função de distância para encontrar os k vizinhos mais próximos de um novo elemento. Com uma regra de classificação, determina qual a classe do novo elemento (para o qual a classe é desconhecida) dada a classe dos seus vizinhos. A ideia é que elementos com características (ou atributos) semelhantes (próximos espacialmente) possuam a mesma classe.

Uma abordagem simplória e funcional para a regra de classificação é a votação majoritária. Este processo determina que o novo elemento pertencerá à classe da maioria dos k vizinhos. A Figura 1 ilustra este processo para um conjuntos de dados com número de classes $C = 2$ (Na Figura definida pela cor e forma) e dimensão[1] $X = 2$ (Na Figura definida pelos eixos X_1 e X_2) .

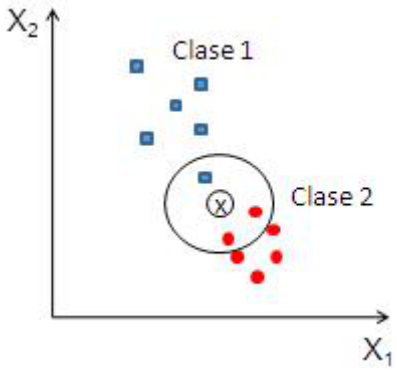

Figura 1: Exemplo do processo de classificação do algoritmo 3NN com $C = 2$ e $X = 2$. Cada elemento é posicionado no espaço de acordo com suas características X_1 e X_2 e sua classe é mapeada na cor e forma do glifo: classe 1 = quadrados azuis, classe 2 = círculos vermelhos. O elemento 'x' (de classe desconhecida), deve ser classificado com base nos 3 vizinhos mais próximos.

[1]Dimensão: cada uma das diversas características (ou atributos) de um dado (altura, largura, peso, comprimento, etc...)

Considere, por exemplo, o algoritmo 3NN (k=3, i.e., os 3 vizinhos mais próximos). A Figura 1 mostra o novo elemento 'x' e o raio que determina os três vizinhos mais próximos. No cenário de votação majoritária, este novo elemento será classificado como pertencente à classe 2 (vermelha) visto que dentre os 3 vizinhos mais próximos 1 elemento pertence à classe 1 e 2 pertencem à classe 2.

3 Arquivos

O seu trabalho deverá ser capaz de criar/manipular dois tipos diferentes de arquivos: arquivo schema (`.schema`) em formato texto e arquivo de dados (`.data`) em formato binário.

3.1 Arquivo Schema (`.schema`)

O arquivo `.schema` contém o formato dos dados que deverão ser processados pelo seu sistema. Ele está salvo em formato textual como o exemplo abaixo.

```
table iris\n
sepal_length double\n
sepal_width double\n
petal_length double\n
petal_width double\n
class char[50]
```

Atenção: A última linha do arquivo `.schema` não possui quebra de linha!

A primeira linha do arquivo schema define o nome do arquivo de dados (também denominado tabela), i.e., no exemplo anterior os dados deverão ser manipulados utilizando o arquivo binário `iris.data`. Observe que todo arquivo de dados terá, portanto, a extensão `.data` adicionada ao nome da tabela presente no arquivo de `.schema`.

As linhas seguintes descrevem cada um dos atributos (ou características) dos dados (Importante ressaltar que o número pode variar de acordo com a aplicação de interesse). O primeiro token contém o nome do atributo e o segundo o tipo (é necessário tratar somente os tipos `int` e `double`).

Por fim, a última linha indica o nome e o tipo da variável que descreve a classe dos elementos. Esta pode ser `int`, `double` ou `char[*]` (em que `*` identifica qualquer número inteiro). Em particular, no nosso exemplo, a classe é determinada pelo atributo com nome `class` e é do tipo `char[50]`.

3.2 Arquivo de Dados (`.data`)

O arquivo de dados (`.data`) é um arquivo binário que contém diversos registros. Cada registro é composto por um **inteiro que o identifica unicamente (atributo `id`), os campos descritos no arquivo textual schema (`.schema`) e um double que armazena uma distância (atributo `dist`)**. Assim, seguindo o exemplo da tabela iris, o arquivo `iris.data` contém os registros como no exemplo abaixo:

```
134         6.3         2.8         5.1         1.5  virginica 0.9432
1           5.1         3.5         1.4         0.2     setosa 0.3423
80          5.7         2.6         3.5         1.0 versicolor 0.5342
23          4.6         3.6         1.0         0.2     setosa 0.3534
```

O arquivo de dados foi exemplificado em formato textual para facilitar a compreensão, mas ele é, de fato, armazenado em um arquivo binário com cada atributo gravado em sequência, i.e., um após o outro. **Não serão aceitos trabalhos em que o arquivo de dados seja textual**. Observe que a primeira coluna é formada pelo atributo `id`, seguidos dos atributos descritos no arquivo `schema` e por fim a campo `dist`.

4 Proposta

Você deverá desenvolver o algoritmo de classificação KNN que, com uma base de dados de teste, é capaz de classificar um novo exemplo. Para isto, ao iniciar, seu programa deverá ler o nome do arquivo que contém o schema dos dados. Para facilitar, o nome do arquivo não terá mais que 30 caracteres e não conterá espaços. Este arquivo textual deverá ser lido pelo seu sistema e interpretado de acordo com a Seção 3.1.

Após o processamento do schema, o seu programa deverá ler do `stdin` todos os registros que compõe a base de dados de treinamento e salvá-los em sequência no arquivo binário de dados (**neste momento utilize `dist` igual** 0.00 **para todos os registros**). Novamente, este arquivo deve ter o nome de acordo com o nome da tabela contida no arquivo schema, i.e., se o arquivo schema descrever a tabela iris, o arquivo binário de dados deverá se chamar `iris.data`. Um exemplo de entrada de um caso de teste é (observem que os dados serão textuais, uma vez que serão encaminhados via `stdin`):

```
iris.schema
43
6.5
3.0
5.2
2.0
virginica
-1
exit
```

Observem que o `id` é o primeiro valor informado e é por meio dele que o seu programa deverá determinar o momento de parar a leitura dos dados de treinamento, i.e., quando o `id` informado for igual a -1 o seu programa deverá encerrar a leitura. Novamente, cada registro deverá ser salvo em sequência em um arquivo binário de dados.

Por fim, após a leitura e processamento, o seu programa deverá executar os comandos abaixo, que serão fornecidos via `stdin`, até o comando de término. Os comandos textuais que seu programa deve obedecer são:

1. `exit`: este comando deverá liberar toda e qualquer memória alocada previamente, fechar possíveis arquivos abertos e por fim finalizar a execução do sistema;
2. `dump_schema`: este comando deverá fazer o dump do arquivo textual schema no `stdout` tal como especificado na Seção 5;
3. `dump_data`: este comando deverá fazer o dump do arquivo binário data no `stdout` tal como especificado na Seção 6;
4. `dump_nn`: este comando deverá fazer o dump dos k vizinhos mais próximos. Maiores detalhes na Seção 7;
5. `knn`: este comando deverá fazer a classificação de um novo registro utilizando o algoritmo KNN. Maiores detalhes na Seção 8;

5 Dump Schema

Este comando tem como objetivo garantir que o seu sistema processou corretamente o arquivo textual `*.schema`. Para isso, ele deve imprimir no `stdout` o conteúdo do arquivo schema da seguinte forma:

```
table <nome_tabela>(<Tamanho_registro> bytes)\n
id int(<tamanho_variavel> bytes)\n
<nome_variavel> <tipo_variavel>(<tamanho_variavel> bytes)\n
...
<nome_variavel> <tipo_variavel>(<tamanho_variavel> bytes)\n
dist double(<tamanho_variavel> bytes)\n
```

A primeira linha deve conter o nome da tabela e o tamanho (em bytes) do registro descrito no arquivo. Em sequência seu sistema deve imprimir o nome dos atributos e o seu respectivo tipo e tamanho (em bytes). Observe que o atributo `id` e `dist`, apesar de não estarem presentes no arquivo schema, estão presentes no dump com seus respectivos tipos e tamanhos. Cada variável deverá ser impressa em um linha única e a ordem deve ser a mesma do arquivo schema. Em nosso exemplo, se usarmos o arquivo schema fornecido na seção 3.1, o output do comando `dump_schema` deverá ser:

```
table iris(94 bytes)\n
id int(4 bytes)\n
sepal_length double(8 bytes)\n
sepal_width double(8 bytes)\n
petal_length double(8 bytes)\n
petal_width double(8 bytes)\n
class char[50](50 bytes)\n
dist double(8 bytes)\n
```

6 Dump Data

Este comando tem como objetivo garantir que o seu sistema criou e processou corretamente o arquivo binário `.data`. Para isso, ele deve imprimir no `stdout` o conteúdo do arquivo data da seguinte forma:

```
<nome_atributo> = <tipo_atributo>\n
...
<nome_atributo> = <tipo_atributo>\n
```

Para cada registro dentro do arquivo de dados você deverá imprimir o nome do atributo (que foi descrito no arquivo schema) e o seu valor. Novamente, apesar de não estarem descritos no arquivo schema, os atributos `id` e `dist` deverão estar presente no output. Em nosso exemplo, se usarmos o arquivo de dados fornecido na seção 3.2, o output do comando `dump_data` seria:

```
id = 134\n
sepal_length = 6.30\n
sepal_width = 2.80\n
petal_length = 5.10\n
petal_width = 1.50\n
class = virginica\n
dist = 0.00\n
id = 1\n
sepal_length = 5.10\n
sepal_width = 3.50\n
petal_length = 1.40\n
petal_width = 0.20\n
class = setosa\n
dist = 0.00\n
...
```

Importante ressaltar que os registros devem ser impressos seguindo a mesma ordem que estão salvos no arquivo binário `.data`. Além disso, quando o valor a ser impresso for do tipo `double`, o seu programa deverá utilizar 2 casas decimais de precisão.

7 Dump Nearest Neighbors

Este comando tem como objetivo garantir que o seu sistema encontrou os k-vizinhos mais próximos de um novo elemento. Para isso, após o comando `dump_nn`, será fornecido ao seu programa (via `stdin`) o valor de k e o novo registro para classificação.

Os dados do novo registro irão ser fornecidos tal como no início do programa com exceção do campo que determina a classe (visto que esta é desconhecida), i.e., serão fornecidos os valores do campo `id` e os valores dos demais campos seguindo a mesma ordem que o arquivo schema determinou. Um exemplo de entrada para o comando `dump_nn` é:

```
dump_nn
3
43
6.5
3.0
5.2
2.0
```

Neste exemplo, deverão ser impresso os 3 vizinhos mais próximos do novo registro com `id` = 43, `sepal_length` = 6.5, `sepal_width` = 3.0, `petal_length` = 5.2 e `petal_width` = 2.0.

Com o registro "em mãos", seus sistema deverá calcular a distância euclidiana dele para todos os outros registros do arquivo data atualizando o campo `dist` de cada um. Observe que os atributos `id` e aquele que descreve a classe (em nosso exemplo `class`) não devem ser levados em consideração no calculo da distância por não proverem nenhuma informação relevante para a classificação.

Por fim, uma vez que o sistema tenha calculado a distância do novo dado para todos os outros, este deverá imprimir os k-vizinhos mais próximos tal como a impressão do comando `dump_data` (seção 6). Novamente, atente para o fato que as alterações nos valores de distância devem ser propagadas para o arquivo data, i.e., você deverá alterar o campo `dist` de cada registro dentro do arquivo de dados.

8 KNN

O comando `knn` deverá executar o algoritmo de classificação para um novo registro que será fornecido via `stdin`. Para isso, após o comando `knn`, será fornecido ao seu programa o valor de k e o novo registro. Os dados do novo registro serão fornecidos tal como no comando `dump_nn` (seção 7), i.e., serão fornecidos os valores do atributo `id` e os valores dos demais atributos seguindo a mesma ordem que o arquivo schema determinou. Um exemplo de input para o comando `knn` é:

```
knn
3
43
6.5
3.0
5.2
2.0
```

Analogamente, após determinado o número de vizinhos k e os atributos do novo registro, deve-se calcular a distância entre o novo registro para todos os outros, atualizando o arquivo de dados com esta nova informação.

Com a distância calculada, o seu sistema deve classificar o novo registro utilizando a regra de classificação por votação majoritária, i.e., deve-se atribuir ao novo ponto a classe que mais aparece nos k vizinhos. Caso ocorra um empate, em particular no nosso trabalho, iremos utilizar os seguintes critérios para o desempate:

1. classificar utilizando 1NN (atribuir a classe do vizinho mais próximo);
2. caso o empate persista atribuir o rótulo da primeira classe (em ordem alfabética).

Após a classificação, o valor calculado para a classe do novo registro deve ser informado por meio do `stdout`. Assim um exemplo de saída deste comando seria:

```
setosa\n
```

Caso a classe seja determinada por um valor do tipo `double` utilize duas casas de precisão para impressão.

9 IMPORTANTE

- Utilize alocação dinâmica (memória heap) e não esqueça de liberar toda memória alocada antes de encerrar a execução do seu programa
- Crie quantas funções achar necessário, a modularização do sistema será levado em consideração no processo de correção.
- **PRAZO FINAL: 04/10 - 23:59:59**