ATAD 2018/19

Algoritmos e Tipos Abstratos de Dados

Enunciado do Projeto

Importação, Normalização e Classificação de Dados

0. Contexto do tema do Projeto

Com o fim de se obter conhecimento clinico útil de pacientes e também de se preverem potenciais doenças, foram registados alguns dados clínicos dos mesmos ao longo de um período de tempo, sendo que um paciente pode estar sem qualquer doença, ou possuir uma de três doenças possíveis.

1. Objetivo do Projeto

Pretende-se desenvolver um programa em C para extrair informação útil de ficheiros com dados sobre os pacientes incluindo alguns dados clínicos, através de um interpretador de comandos. Esta capacidade engloba a maioria do projeto.

Adicionalmente, como uma funcionalidade mais avançada, pretende-se fazer a previsão da existência de doenças em pacientes tendo em conta a sua idade e alguns dados clínicos, usando redes neuronais. Neste caso a previsão reduz-se à existência de uma das 3 possíveis doenças, ou nenhuma, num dado paciente.

1.1 Representação de um paciente em Memória

Cada paciente é representado, obrigatoriamente, pela estrutura de dados apresentada na Figura 1, sendo *Date* um tipo de dados apropriado para guardar uma data.

```
#define DOENCA1    1
#define DOENCA2    2
#define DOENCA3    3
#define SEM_DOENCA 4

typedef struct clinicalData {
   int age;        /* Média da idade em anos quando os dados foram obtidos */
   float bmi;      /* Valores médios do Imc- Indice Massa corporal médio (Kg/m2) */
   float glucose;  /* Valores médios da Glicose (mg/dl) */
   float insulin;  /* Valores médios da Insulina  (µU/ml) */
   float mcp1;     /* Valores médios da Proteina MCP-1 (pg/ml)*/
   int disease_type /* Classificacao do Doente 1 – Doença1  2 – Doença2
                    /*  3 – Doença3  4 – Sem Doença
                      Usado apenas na opcao NEURALNET*/

   int clinicalDataCount /* Numero de vezes que os dados foram obtidos */

    /* pode acrescentar algum atributo/campo se achar relevante */
} ClinicalData;
```

```
typedef struct patient {
    int id;                     /*Número que identifica um paciente */
    Date birthdate;             /*Data de nascimento*/
    char gender;                /*Género (M ou F)*/
    String hospital;            /*Nome do hospital de referencia do paciente*/
    String district;            /*Distrito onde pertence o hospital*/
    ClinicalData clinicalData;  /*Dados clínicos do doente (médias)*/
} Patient;
```

Figura 1 – Tipo de dados Patient.

Na implementação dos comandos descritos neste enunciado podem definir/utilizar outros tipos de dados auxiliares que achem úteis para a resolução dos problemas.

1.2 Dados de entrada

Existem três tipos de ficheiros com dados:

- Ficheiro de dados sobre os pacientes;
- Ficheiros de dados sobre com os dados clinicos dos pacientes.
- Ficheiros de dados de treino sobre com os dados clínicos dos pacientes, usado no comando NEURALNET.

Todos os ficheiros se encontram em formato CSV.

Ficheiro dos pacientes (cada linha corresponde a informação sobre um paciente)

```
<id_patient>; <birth_date>; <gender>; <hospital>; <district>
...
```

Ficheiro com os dados clinicos dos pacientes

```
<id_patient>; <date>; <age>; <bmi>; <glucose>; <insulin>; <mcp1>;
...
```

Ficheiro com os dados clínicos de treino dos pacientes

```
<id_patient>; <date>; <age>; <bmi>; <glucose>; <insulin>; <mcp1>; <type_disease>;
<c1>; <c2>; <c3>; <c4>
...
```

O valor *<birth_date>* encontra-se no formato "dd/mm/aa".
Os campos <type_disease>, <c1>,<c2>, <c3> e <c4> são usados no treino das Redes Neuronais

Pode-se assumir que não existem ficheiros "mal-formados".

Neste projeto os ficheiros serão lidos aos pares e a informação composta desses ficheiros deve ser guardada nos tipos de dados definidas em 1.1.

1.2.1 Ficheiros Disponibilizados

Juntamente com este enunciado são disponibilizados no moodle, 4 ficheiros de entrada para testes:

- patients.csv
- clinicalData.csv
- patients_train.csv
- clinicalData_train.csv

Os dois últimos ficheiros são os ficheiros com os dados de treino (i.e., para o comando NEURALNET).

Será disponibilizado também um ficheiro fonte main.c que inclui o interpretador de comandos do menu principal.

Após a divulgação do enunciado será disponibilizado no Moodle um exemplo dos resultados esperados na execução dos vários comandos da aplicação para estes ficheiros.

1.3 Utilização de TADs

É obrigatória a manutenção em memória da informação importada exclusivamente numa instância do TAD List, sendo ListElem o tipo de dados definido em 1.1.

Cada uma das operações detalhadas na secção seguinte poderá indicar a obrigatoriedade de criação de novas instâncias de TAD List e/ou TAD Map.

A não aderência a estas obrigatoriedades invalida a cotação total das implementações correspondentes.

Não é permitido alterar os comportamentos lecionados dos TAD, nomeadamente os header files dos TAD stack, queue, list e map.

1.4 Comandos

Cada comando é representado por uma palavra que pode ser escrita pelo utilizador em maiúsculas ou em minúsculas. **Para todos os comandos, exceto LOAD, caso não existam dados carregados em memória (não lidos ou apagados), o comando deve acusar na consola "SEM DADOS CARREGADOS"**.

Os comandos são os seguintes:

- **LOAD**

  Importa a informação de um par de ficheiros cujos nomes são solicitados ao utilizador (e.g., patients.csv e clinicalData.csv). Ambos têm de existir; caso algum não exista a operação não é executada.

Após o comando, deverá ser apresentado na consola uma mensagem, e.g., **"Foram lidos <N> pacientes e informação sobre <M> dados clinicos"**

A importação dos dados deve ter em conta o especificado em 1.1, sendo calculados valores médios na estrutura *Patient* (no atributo *clinicalData*) respeitantes aos dados clinicos de cada paciente.

O cálculo de cada valor médio deve ser incremental, e pode ser calculado de acordo com a seguinte fórmula:

$$avg_{n+1} = \frac{avg_n * n + v_{n+1}}{n+1} \qquad com\ avg_0 = 0$$

onde avg_n é a média com n elementos e v_n o elemento na posição n.
Por exemplo se o valor médio com dois valores for igual a 20, e se adicionar-mos o valor 30, o valor médio com três valores será

$$avg_3 = \frac{avg_2 * 2 + v_3}{3} = \frac{70}{3} = 23{,}3(3)$$

Após a importação, os dados lidos nunca devem ser modificados, utilizará cópias dos dados para qualquer manipulação necessária (isto é descrito nos próximos comandos).

- **CLEAR**

Limpa a informação atualmente em memória. Deverá indicar o número de registos que foram descartados, e.g., **"Foram apagados <N> registos de pacientes"**

- **SHOW**

Mostra a informação dos pacientes no formato em que ela é guardada em memória.

- **QUIT**

Termina a aplicação e liberta qualquer memória ainda alocada.

- **SORT**

Mostra os pacientes importados de forma ordenada crescentemente. O critério de ordenação deve ser solicitado ao utilizador e pode ser um dos seguintes:

- Data de nascimento
- Hospital (desempate pela data de nascimento)
- Distrito (desempate pelo hospital);

Na implementação do comando, todos os dados deverão ser copiados para uma instância auxiliar do TAD List e a ordenação deverá ser efetuada sobre esta instância; os resultados deverão ser mostrados com a operação *listPrint* do TAD List.

- **AVG**

Mostra a média dos dados clinicos de cada paciente (age, bmi, glucose, insulin e mcp1) em cada distrito, e ordenados por distrito. Use a função averageClinicalData(PtList patients) para esse efeito. AVG baseia-se no resultado do comando LOAD. Para guardar os resultados, utilize o mesmo tipo de dados que aquele que for utilizar na opção CHECKDISTRICT.

- **NORM**

Mostra, para cada paciente, os seus dados clínicos normalizados entre -k e k, segundo a normalização *min-max*.

Cada valor é calculado por $x' = \frac{x - min_i}{max_i - min_i} * 2k - k$,sendo min_ie max_i os valores do mínimo e máximo de cada dado clinico *i* entre todos os pacientes, i.e., os valores máximo/mínimo de *cada* dado clínico entre todos os pacientes permitem normalizar os seus valores individuais num dado intervalo pretendido.

A implementação deste comando deverá utilizar uma função auxiliar

```
PtList normalizeClinicalData(PtList patients)
```

recebe a lista de pacientes obtida no comando LOAD e devolve uma nova instância com cópias dos pacientes, mas cujas estatísticas estão normalizadas no intervalo [-k,k].
O valor de k deve ser pedido ao utilizador.

- **QUEUE**

Este comando deverá copiar para uma instância do TAD Queue todos os pacientes que se enquadrem nos seguintes critérios:

-A sua idade seja inferior ao valor médio do intervalo [min(age), max(age)] das idades de todos os pacientes.

ou

-A sua idade seja superior ao valor médio do intervalo [min(age), max(age)] das idades de todos os pacientes.

-O valor dos atributos bmi, glucose, insuline e mcp1 sejam inferiores ao valor médio do intervalo entre o [min(atr),max(atr)] para cada um destes 4 atributros.

Após este procedimento, deverá, repetidamente, solicitar um dos seguintes comandos ao utilizador:

next – retira um elemento da fila e mostra a informação do paciente respetivo; caso a fila se encontre vazia deve-se informar o utlizador.
stop – termina este processo de repetição.

- **CHECKDISTRICT**

Segmenta os pacientes de acordo com o seu distrito de residência, usando uma TAD Map para mapear o distrito do paciente para informação de pacientes sobre esse distrito.
O "valor" mapeado deve ser de um tipo tal (definido pelo(s) aluno(s)) que permita guardar:

- district;
- avgAgeDistrict /* media das idades no distrito */
- avgBMIDistrict /* media dos valores de IMC no distrito */
- avgGlucoseDistrict /* media dos valores de glicose no distrito */
- avgInsulinDistrict /* media dos valores de insulina no distrito */
- avgMcp1District /* media dos valores da proteina MCP1 no distrito */

O programa deve então proceder à solicitação do distrito (chave do mapa) e mostrar o valor associado; isto até que seja introduzido um distrito vazio (distrito igual à string “”).

A pesquisa da informação associada deve ser feita exclusivamente através das operações do TAD Map.

- **LOADT**

Importa a informação de um par de ficheiros de treino usados na opção NEURALNET, cujo os nomes são solicitados ao utilizador (e.g., patients_train.csv e clinical_train.csv). Ambos têm de existir; caso algum não exista a operação não é executada.

- **NORMT**

Comando semelhante ao comando NORM, recebendo neste comando a lista de pacientes obtida no comando LOADT.
Deverá usar o valor k=5. Os seus resultados serão usado no comando NEURALNET.

- NEURALNET

Permite classificar os pacientes em 4 grupos de acordo com os seus dados clinicos, usando os algoritmos de Redes Neurais (ver Anexo I). As classificações possíveis serão:

1- Doença1
2- Doença2
3- Doença3
4- Sem doença

Deve usar uma rede neuronal com 5 entradas, 5 neurónios na camada escondida, 4 neurónios na camada de saída, e cuja função de ativação é a função sigmóide.
A entrada deve ter em conta os seguintes dados clínicos:

a. age
b. bmi
c. glucose
d. insulin
e. mcp1

A classificação de cada paciente a partir da rede neuronal será um valor real entre 0 e 1 para cada uma das saídas. Sendo a posição da saída com maior valor aquela que corresponderá à classificação do paciente.
Ex: Se saida = [0.9621 0.012 0.008 0.003] a classificação do paciente será 1(Doença1), se saída = [0.0235 0.0123 0.9738 0.009] a classificação do paciente será 3 (Doença3).

Nota: Este algoritmo deve operar com dados dos pacientes já normalizados, obtidos a partir da função auxiliar *normalizeClinicalData* dos comandos NORM e NORMT.

a) Execute a rede neuronal usando valores fixos para todos os pesos, faça por exemplo $w_{ij} = 0,5$. Compare as saidas desta rede com os dados de treino lidos no comando LOADT e normalizados pelo comando NORMT, usando estes últimos como entrada na rede.
b) Execute uma rede neuronal previamente treinada com os dados de treino usando o algoritmo de retropropagação. Use como entrada os dados os dados obtidos dos pacientes lidos no comando LOAD e normalizados no comando NORM.
   Mostre em cada linha os dados clínicos do paciente e a sua classificação.

   Os dados da rede treinada serão fornecidos num ficheiro *neural_net.txt*, que contém o número de entradas, o número de neurónios na camada escondida, o número de saídas, e cada um dos pesos que chega a cada um dos 9 neurónios (5+4).

2 Relatório

No relatório deverão constar as seguintes secções (para além de capa com identificação dos alunos e índice):

a) Descrição dos TAD utilizados, descrição da implementação e justificação da estrutura de dados escolhida para a implementação;
b) Para cada comando (exceto CLEAR,SHOW e QUIT) fornecer:
   - A complexidade algorítmica da respetiva implementação, justificada com trechos de código respetivo à função
c) Limitações: Quais os comandos que apresentam problemas ou não foram implementados;
d) Conclusões: Análise crítica do trabalho desenvolvido.

3 Tabela de Cotações e Penalizações

A avaliação do trabalho será feita de acordo com os seguintes princípios:

- Estruturação: o programa deve estar estruturado de uma forma modular e procedimental;
- Correção: o programa deve executar as funcionalidades, tal como pedido.
- Legibilidade e documentação: o código deve ser escrito, formatado e comentado de acordo com o standard de programação definido para a disciplina.

A nota final obtida, cuja tabela de cotações se apresenta a seguir, será ponderada de acordo com os princípios acima descritos.

Funcionalidade/Comando	**Cotação (val)**
LOAD + LOADT +SHOW + CLEAR + QUIT	4
SORT	2
AVG, AVGT	2
NORM, NORMT	2
QUEUE	2,5
CHECKDISTRICT	2,5
NEURALNET	
Execução de uma rede neuronal não treinada	1,5
Execução de uma rede neuronal treinada	1,5
Relatório	2
TOTAL	20 valores

A seguinte tabela contém penalizações a aplicar:

Descrição	**Penalização (val)**
Uso de variáveis globais	até 2
Não separação de funcionalidades em funções/módulos	até 3
Não libertação de memória	até 3
Não utilização dos TAD obrigatórios	anulado

4 Instruções e Regras Finais

O IDE a utilizar fica ao critério dos alunos, mas, caso não utilizem o IDE usado na disciplina (i.e., Visual Studio), terão que, **antes de submeter, criar os respetivos projetos finais no IDE** Visual Studio 2017.

O não cumprimento das regras a seguir descritas implica uma penalização na nota do trabalho prático. Se ocorrer alguma situação não prevista nas regras a seguir expostas, essa ocorrência deverá ser comunicada ao respetivo docente de laboratório de ATAD.

Regras:

a) O Projeto deverá ser elaborado por **dois alunos do mesmo docente de laboratório**.

b) A nota do Projeto será atribuída individualmente a cada um dos elementos do grupo após a discussão. As discussões possuem uma componente oral de apresentação e validação do projeto e uma componente prática individual.

c) A apresentação de relatórios ou implementações plagiadas leva à imediata atribuição de nota zero a todos os trabalhos com semelhanças, quer tenham sido o original ou a cópia.

d) No rosto do relatório e nos ficheiros de implementação deverá constar o número, nome e turma dos autores e o nome do docente de laboratório a que se destina.

e) O trabalho deverá ser submetido no moodle, no link do respetivo docente de laboratórios criado para o efeito, até às **11:00 do dia 14 de Junho**. Para tal terão que criar uma pasta com o nome: **nomeAluno1_númeroAluno1-nomeAluno2_númeroAluno2**, onde colocarão o ficheiro do relatório em formato **pdf** e uma pasta com o projeto Visual Studio da implementação das aplicações a desenvolver. Os alunos terão de submeter essa **pasta compactada em formato ZIP**. Apenas será permitido submeter um ficheiro.

f) Não serão aceites trabalhos entregues que não cumpram na íntegra o ponto anterior.

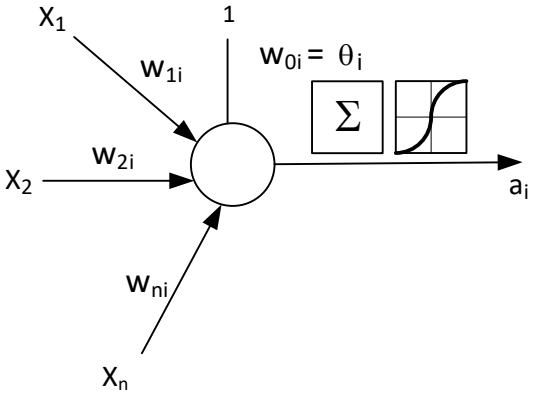
g) As datas das discussões serão publicadas após a entrega dos trabalhos.

Anexo I

Redes Neuronais

As redes neuronais são algoritmos que pretendem simular o funcionamento dos neurónios do cérebro (https://en.wikipedia.org/wiki/Neural_network). Uma das suas aplicações mais usuais é a classificação de dados e a simulação de funções.

O funcionamento das redes baseia-se no neurónio artificial, que recebe n entradas e devolve uma saída. Sendo a saída o resultado da aplicação do somatário do produto de cada entrada X_i com o respetivo peso (w_{1i}), seguida da aplicação do resultado deste somatório a uma função de ativação f. Um exemplo de uma função de ativação é a função sigmóide $f(x) = \frac{1}{1+e^{-x}}$. Considera-se o peso W_{0i} como sendo o *bias*, cujo entrada é sempre fixa com o valor 1.



com $a_i = f(\sum_{j=0}^{n}(w_{ji} * x_j))$

É importante referir que a função sigmóide tem como contradomínio (valores possíveis) o intervalo [0,1], ou seja o seu resultado pode ser visto como uma probabilidade de uma dada característica existir. Este fato é bastante importante quando estamos a usar a rede neuronal como um classificador.

Os neurónios podem ser juntos em rede em várias camadas ligadas entre si, formando uma rede neuronal:

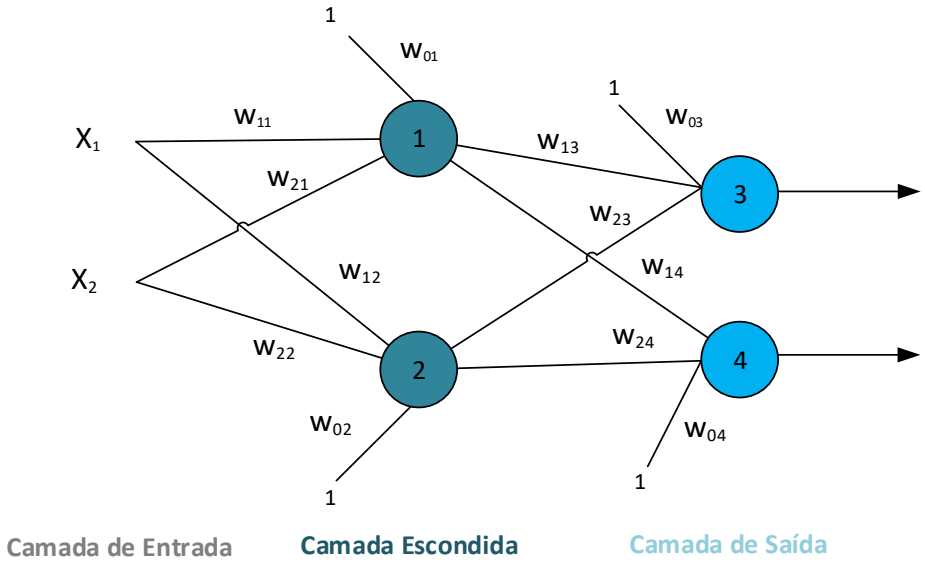
- As camadas escondidas onde cada camada pode ter vários neurónios.
- As camadas de saída de onde saiem os valores de saída da rede.
- Costuma-se designar camada de entrada àquela que, apesar de não conter neurónios, contém os valores de entrada da rede.

As redes neuronais armazenam nos seus pesos os valores necessários à simulação de funções ou à classificação, funcionando como uma caixa negra (*black box*) com n entradas e m saídas.

Existem vários algoritmos para treinar uma rede (atravé da alteração dos seus pesos), partindo de dados de treino. Um dos algoritmos mais usados nas redes neuronais com camadas escondidas é o algoritmo de **retropropagação**.

Depois de treinada, a rede deve responder (usando dados de teste) a entradas com características similares às dos dados de treino com os valores das saidas equivalentes aos existentes nos dados de teste.

Na figura seguinte, dá pra ver um exemplo de uma rede com duas entradas e duas saídas. No meio existe uma camada escondida com 2 neurónios.



No recurso:

· http://web.tecnico.ulisboa.pt/ana.freitas/bioinformatics.ath.cx/bioinformatics.ath.cx/index9bb2.html?id=111

pode visualizar-se, usando uma aplicação flash, a execução de uma rede neuronal multi-camada, com 2 entradas, uma camada escondida com 2 neurónios e uma saída. Esta aplicação mostra também a aplicação do algoritmo (de aprendizagem) de retropropagação.

(fim de enunciado)