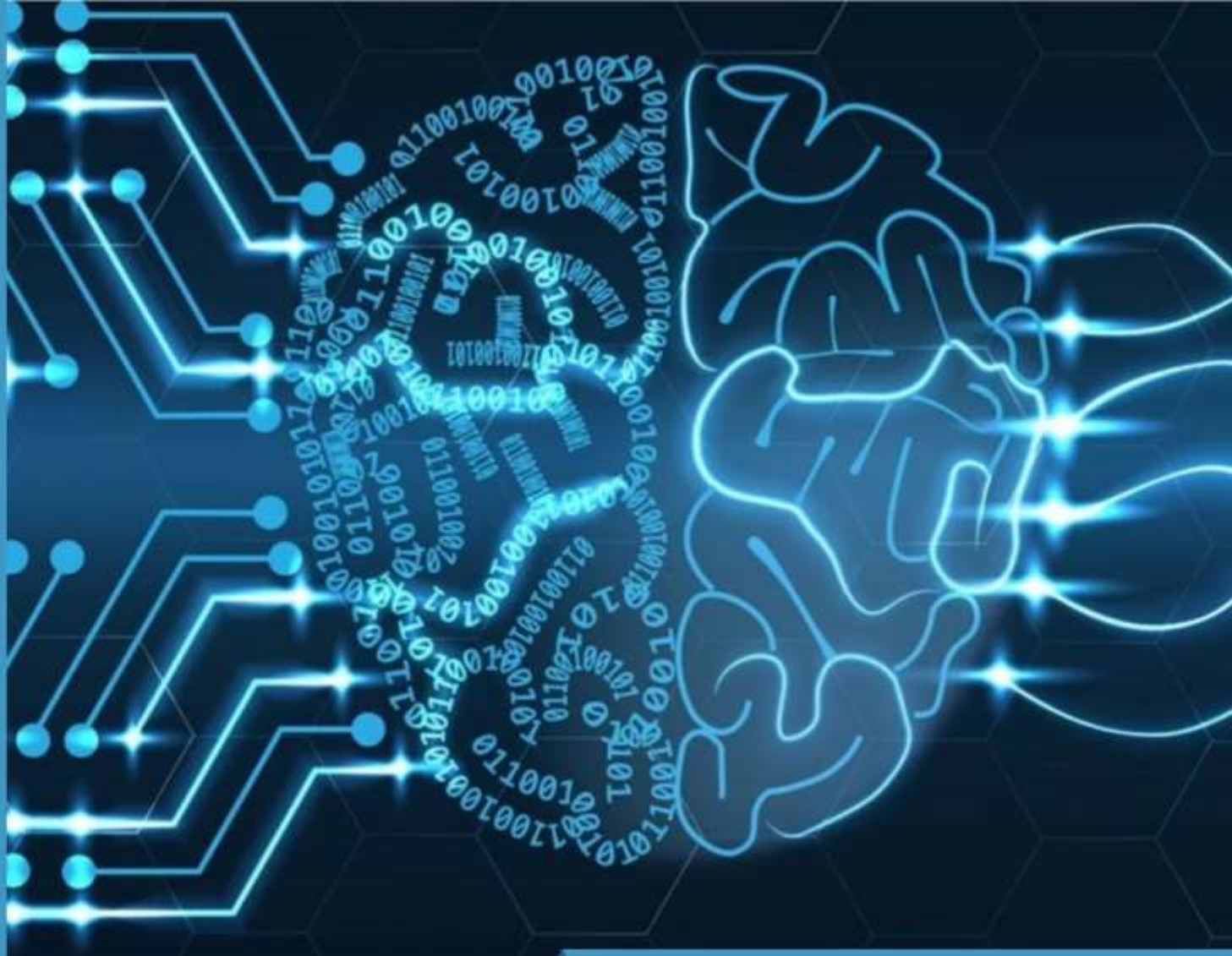
ENGENHARIA INFORMÁTICA



# TRABALHO PRÁTICO

## CONHECIMENTO E RACIOCÍNIO

RAMO DESENVOLVIMENTO DE APLICAÇÕES

DIOGO OLIVEIRA
202146036

LARA BIZARRO
2021130066

MAIO,2024

# Índice

# Índice de Figuras

# Resumo

No âmbito da Licenciatura em Engenharia de Informática, para a cadeira Conhecimento e Raciocínio (CR), mais especificamente, no ano letivo 2023/2024, foi proposto aos alunos a escolha, estudo, treino e análise de redes neuronais. O enfoque do enunciado reside na exploração e aprofundamento dos conceitos de raciocínio baseado em casos e redes neurais no contexto do diagnóstico médico de doenças hepáticas, como hepatite, acidente vascular cerebral (AVC) e cirrose. Os alunos foram desafiados a selecionar um conjunto de dados entre três opções disponíveis: *Hepatitis*, *Stroke* ou *Cirrosis*.

# Introdução

Partindo da preparação do conjunto de dados, que implica a conversão de atributos em valores numéricos ou booleanos e o tratamento de valores em falta, exploramos o uso de raciocínio baseado em casos para preencher esses valores ausentes. Posteriormente, avançamos para o estudo e análise de redes neurais, utilizando diferentes arquiteturas, funções de treino e de ativação, e avaliando a capacidade de generalização e desempenho das redes utilizando conjuntos de dados de teste. Com isto decidimos escolher o *dataset* referente à hepatite.

# Hepatite

## Porquê a hepatite?

Escolhemos a hepatite para este trabalho por várias razões.

Primeiramente, a hepatite é uma doença hepática comum e grave que afeta milhões de pessoas em todo o mundo, representando um problema significativo de saúde pública.

Além disso, a hepatite apresenta uma variedade de sintomas e fatores de risco que a tornam um desafio interessante para o desenvolvimento e aplicação de técnicas de raciocínio baseado em casos e redes neurais. A capacidade de identificar e classificar corretamente os casos de hepatite com base em dados clínicos e laboratoriais é crucial para um diagnóstico preciso e um tratamento eficaz.

Portanto, a escolha da hepatite como tema central deste trabalho é fundamentada na sua relevância social e nos desafios diagnósticos.

## O que é a hepatite?

O fígado desempenha diversas funções essenciais no organismo humano, desde a digestão até o armazenamento de energia e a remoção de toxinas.

A hepatite é uma inflamação do fígado e pode ter várias origens, sendo os vírus as causas mais comuns. Quando ocorre, compromete a capacidade do fígado a realizar as suas funções, podendo resultar em danos hepáticos progressivos, como cirrose ou cancro. A gravidade da doença varia consideravelmente, desde formas leves que

se resolvem com repouso até casos mais graves que exigem tratamentos complexos para controlar a progressão da doença.

Dentro das hepatites virais, existem seis tipos principais de vírus: A, B, C, D, E.

As hepatites A e E são geralmente adquiridas através da ingestão de água ou alimentos contaminados. Por outro lado, as hepatites B, C e D são transmitidas através do contato com o sangue contaminado, seja por via sexual, transfusões de sangue, compartilhamento de seringas ou transmissão de mãe para filho durante a gravidez. As formas virais são responsáveis por um grande número de casos em todo o mundo, sendo comuns as formas crônicas nas hepatites B e C.

Em Portugal, a hepatite A é relativamente comum, afetando principalmente crianças e adultos jovens. Geralmente, resolve-se em poucas semanas sem deixar sequelas permanentes. A hepatite B é considerada uma das mais graves, podendo tornar-se crônica e evoluir para cancro hepático. Afeta milhões de pessoas globalmente e é transmitida principalmente por contato sexual e compartilhamento de agulhas. Já a hepatite C, que frequentemente evolui para formas crónicas, é uma das principais causas de cirrose e cancro hepático em Portugal.

A hepatite D ocorre apenas em pacientes infetados com o vírus da hepatite B, aumentando a gravidade da infeção. A hepatite E é transmitida através da ingestão de água ou alimentos contaminados e, geralmente, não se torna crônica. Embora ainda não haja uma vacina definitiva contra a hepatite C, o desenvolvimento de novas terapias está em andamento.

## Sintomas e causas

A maioria das vezes, as hepatites não apresentam sintomas visíveis. Quando ocorrem, os sintomas mais comuns incluem fadiga, perda de apetite, náuseas, vômitos, diarreia, urina escura, fezes claras, dores abdominais e icterícia (coloração amarela da pele e dos olhos). Uma hepatite pode se tornar crônica e progredir para danos mais graves no fígado, como cirrose ou câncer hepático, se persistir por mais de seis meses. Todos os tipos de hepatite requerem avaliação médica e acompanhamento adequado.

As causas das hepatites podem ser diversas, incluindo agentes infeciosos, como bactérias ou vírus, bem como o consumo de álcool, medicamentos e algumas plantas. Além disso, existem as hepatites autoimunes, em que o sistema imunológico ataca as células do fígado. Este tipo afeta principalmente mulheres, entre 20 e 30 anos, e entre 40 e 60 anos.

# Tratamento de Dados

O *dataset* referente à hepatite inclui informações sobre albumina (ALB), fosfatase alcalina (ALP), alanina aminotransferase (ALT), aspartato aminotransferase (AST), bilirrubina (BIL), colinesterase (CHE), colesterol (CHOL), creatinina (CREA), gama-glutamil transferase (GGT) e proteína total (PROT). Os atributos *Category* (*target*) representam diferentes condições de saúde, enquanto *Sex* indica o género do indivíduo, quanto ao *ID* é uma coluna ignorada uma vez que não representa um dado de relevância para este trabalho.

Para o melhor entendimento do *dataset* fornecido foi feito uma pesquisa profunda, tanto para o entendimento geral do que é a hepatite como a importância dos dados mencionados acima para o diagnóstico da mesma, os *websites* consultados mais pertinentes encontram-se nas referências bibliográficas (entre eles pode encontrar páginas da *SNS24* e publicações do *google scholar*).

## Análise e identificação dos dados mais relevantes

Em primeiro lugar, vamos explicar o significado de cada um dos dados, especialmente aqueles cujo nome não é explícito.

A **albumina** (ALB) é uma proteína produzida pelo fígado que ajuda a manter a *pressão oncótica*[1] no sangue e transporta várias substâncias, incluindo hormonas, medicamentos e ácidos gordos essenciais. Níveis baixos de albumina podem indicar dano ou

[1] **Pressão oncótica** - pressão osmótica exercida pelas proteínas plasmáticas, principalmente pela albumina. É responsável por atrair a água à volta para a corrente sanguínea dos tecidos circundantes, ajudando a manter o equilíbrio adequado de fluidos nos capilares sanguíneos.

disfunção hepática, desnutrição, doença renal ou outros problemas de saúde.

A **fosfatase alcalina** (ALP) é uma enzima encontrada em vários tecidos do corpo, incluindo o fígado, ossos, rins e intestinos. Níveis elevados estão frequentemente associados a distúrbios hepáticos ou ósseos. No contexto da doença hepática, níveis elevados de ALP podem indicar obstrução dos *ductos biliares*[2], tumores hepáticos ou certas condições hepáticas.

**Alanina Aminotransferase** (ALT) é uma enzima encontrada principalmente no fígado, desempenha um papel fundamental no metabolismo dos aminoácidos. Níveis elevados no sangue são frequentemente indicativos de dano ou inflamação hepática, observados em condições como hepatite, doença hepática gordurosa ou lesão hepática devido a drogas ou álcool.

**Aspartato Aminotransferase** (AST) é uma enzima encontrada em vários tecidos, incluindo o fígado, coração, músculos e rins. Assim como a ALT, a AST também está envolvida no metabolismo dos aminoácidos. Níveis elevados de AST podem ser um sinal de dano hepático, embora a AST seja menos específica, também pode ser observado em ataques cardíacos, lesões musculares e outras condições.

**Bilirrubina** (BIL) é um produto de resíduos produzido quando o fígado decompõe os glóbulos vermelhos velhos. Níveis elevados podem resultar de doenças hepáticas como hepatite, cirrose ou obstrução dos *ductos biliares*.

**Colinesterase** (CHE) é uma enzima que desempenha um papel na quebra da acetilcolina, um neurotransmissor. Os níveis de

[2] **Ductos biliares** – série de canais que transportam a bile (líquido produzido pelo fígado) para o intestino delgado.

colinesterase podem estar diminuídos em doenças hepáticas ou certas condições genéticas.

**Colesterol** (CHOL) é um tipo de gordura encontrada no sangue. Níveis anormais podem indicar disfunção hepática, especialmente em condições como doença hepática gordurosa.

**Creatinina** (CREA) é um produto de resíduos produzido pelo metabolismo muscular. Embora não esteja diretamente relacionada à função hepática, os níveis de creatinina são frequentemente medidos em testes de função hepática para avaliar a função renal, já que ambos os órgãos estão intimamente ligados.

**Gama-Glutamil Transferase** (GGT) é uma enzima encontrada no fígado, *ductos biliares* e outros tecidos. Níveis elevados estão frequentemente associados a doenças hepáticas, especialmente aquelas envolvendo obstrução dos ductos biliares ou abuso de álcool.

**Proteína Total** (PROT) mede a quantidade total de proteína no sangue, incluindo albumina e outras proteínas. Níveis anormais podem indicar doença hepática, desnutrição, doença renal ou outros problemas de saúde.

Com este pequeno resumo do que cada um destes dados significa e tendo em conta os outros dados que não precisam de explicação chegamos à conclusão de que **os dados mais importantes** eram **ALT, AST, BIL, ALB e ALP**.

Para chegarmos a uma conclusão tivemos em atenção os dados que eram mais específicos da doença hepatite, uma vez que alguns dados são comuns a outras doenças ou menos discriminantes da hepatite.

Após a análise da importância do sexo e da idade da pessoa para o diagnóstico de hepatite chegamos à seguinte lista (em que o **1 é o mais importante** e o **7 é o menos importante** para **diagnosticar hepatite**):

1. **ALT**
2. **AST**
3. **BIL**
4. **ALB e ALP**
5. **CHE**
6. **CHOL, CREA, GGT e PROT**
7. ***Age* e *Sex*.**

Considerando que a lista é relativa, ou seja, os dados foram comparados entre si em termos de importância, segue-se a explicação da categorização anterior:

Como referido anteriormente **ALT** é o dado considerado mais importante, uma vez que está presente em grande quantidade no fígado. Quando o fígado está danificado ou inflamado é libertada na corrente sanguínea, além disso, é comum em muitos tipos de hepatite.

A **AST** também é muito importante, contudo pode ser encontrada noutros tecidos para além do fígado, o que a torna ligeiramente menos especifica.

O aumento da **BIL** indica disfunção hepática, sendo um marcador importante para *icterícia*[3], um sintoma comum na hepatite. No entanto, não é exclusivo da hepatite, pode ser um indicador da cirrose.

[3] **Icterícia** – Condição médica em que a pele e as mucosas adquirem uma coloração amarelada devido ao acumulo de BIL

A **ALB** é considerada um marcador importante, uma vez que é sintetizada pelo fígado e sua *concentração sérica*[4] pode ser afetada em casos de hepatite. Todavia, em termos de importância relativa como indicador específico para hepatite, variações na sua *concentração sérica* podem não ser tão específicas para hepatite como as alterações nas enzimas hepáticas ou na BIL.

Quanto a **ALP** à semelhança da ALB é importante, no entanto, não é um indicador tão específico, uma vez que pode indicar *obstrução biliar*[5].

**CHE** embora menos específica, é produzida pelo fígado por isso pode ser um indicador do estado do fígado.

Quanto ao CHOL, à CREA, à GGT e à PROT, apesar de mais relevantes que a idade e o sexo, são as menos importante para diagnosticar hepatite, visto que não são indicadores específicos da doença e a sua variação também pode estar associada a outras doenças (por exemplo, CREA é um indicador específico da função renal).

A **idade** apesar de importante na manifestação da doença (afeta os sintomas), não é significativa para o diagnóstico. Por fim, o **sexo** não é de todo importante, sendo que afeta mais na maneira como se pode apanhar a doença.

[4] **Concentração sérica** - quantidade de uma substância específica presente no sangue.
[5] **Obstrução biliar** - é uma condição médica em que há bloqueio ou obstrução do fluxo de bile nos ductos biliares.

# Case Based Reasoning (CBR)

De forma a completar os dados em falta (NA) do ficheiro *Train* (Excel) foi aplicado o CBR, que, neste caso, tem em conta o conhecimento acumulado dos casos anteriores para preencher os dados dos casos em falta. Simplificando, esta abordagem procura os casos anteriores semelhantes ao caso que tem dados em falta e, substitui os dados em falta pelos valores dos dados do caso anterior mais idêntico.

## *A nossa abordagem*

Antes de falarmos do CBR é importante falar da preparação do mesmo, presente no ficheiro *TP.m*[6].

Nesta função é lido o ficheiro Excel (*Train*) para a variável *case_library*, posteriormente o *array* é corrido num ciclo de forma a substituir o conteúdo da *category* por um número (o que antes se apresentava no formato "*0=Blood Donor*" passa a ser apenas 0), se a categoria não fizer parte das 5 categorias dadas, esta passa a ter um espaço (nomeadamente as zonas preenchidas com NA). Também a coluna *Sex* é modificada, para 0 se for masculino e para 1 se for feminino. Após o ciclo converte-se tudo a *double*, de forma a ficarem valores numéricos e não *strings*. Por fim, faz-se outro ciclo de forma a detetar quais os casos que apresentam o espaço vazio, esses são copiados para uma nova variável *new_case;* depois é chamada a função *retrieve* para identificar quais são os casos com valores semelhantes usando o CBR.

[6] **Nota** - A função TP apenas é explicada até à chamada da função *retreive*, a sua continuação estará no final da explicação do CBR com a nota 7.

```
similarity_threshold = 0.9;
%formato de leitura das colunas do excel

formatSpec = '%f%s%f%s%f%f%f%f%f%f%f%f%f%f';

case_library = readtable('Train.csv', ...
            'Delimiter', '\t', ...
            "TextType", "string", ...
            'Format', formatSpec);
```

*Figura 1 - Threshold, Formato esperado e Leitura do Ficheiro Excel*

```
for i=1:height(case_library)
    category_text_tipo = case_library.Category(i);

    if strcmp(category_text_tipo, '0=Blood Donor')
        case_library.Category(i) = 0;
    elseif strcmp(category_text_tipo, '0s=suspect Blood Donor')
        case_library.Category(i) = 1;
    elseif strcmp(category_text_tipo, '1=Hepatitis')
        case_library.Category(i) = 2;
    elseif strcmp(category_text_tipo, '2=Fibrosis')
        case_library.Category(i) = 3;
    elseif strcmp(category_text_tipo, '3=Cirrhosis')
        case_library.Category(i) = 4;
    else
        case_library.Category(i) = ' ';
    end

    if case_library.Sex(i) == 'm'
        case_library.Sex(i) = 0;
    elseif case_library.Sex(i) == 'f'
        case_library.Sex(i) = 1;
    end
end
case_library.Sex = str2double(case_library.Sex);
```

*Figura 2 - Substituição de valores*

```
for i=1:height(case_library)
    if strcmp(case_library.Category(i), ' ')
        new_case.ID = case_library.ID(i);
        new_case.Category = ' ';
        new_case.Age = case_library.Age(i);
        new_case.Sex = case_library.Sex(i);
        new_case.ALB = case_library.ALB(i);
        new_case.ALP = case_library.ALP(i);
        new_case.ALT = case_library.ALT(i);
        new_case.AST = case_library.AST(i);
        new_case.BIL = case_library.BIL(i);
        new_case.CHE = case_library.CHE(i);
        new_case.CHOL = case_library.CHOL(i);
        new_case.CREA = case_library.CREA(i);
        new_case.GGT = case_library.GGT(i);
        new_case.PROT = case_library.PROT(i);

        [retrieved_indexes, similarities, new_case] = retrieve(case_library, new_case, similarity_threshold, i);
```

*Figura 3 - Copia do caso vazio para uma variável*

A função referente ao CBR está presente no ficheiro retreive.m, esta função recebe os valores do ficheiro (*case_library*), o caso com espaço vazio que se pretende preencher (*new_case*), o limite para impedir o CBR de considerar casos abaixo de 0.9 de similaridade (*threshold*) e a posição do caso com espaço vazio (*pos_new_case*) e devolve os índices dos casos mais similares (retreived_indexes), o quão similar é cada caso (*similarities*) e o caso com espaço em branco (*new_case*).

```
function [retrieved_indexes, similarities, new_case] = retrieve(case_library, new_case, threshold, pos_new_case)
```

*Figura 4 - Cabeçalho da função retrieve*

Sabendo que é na coluna ***Category*** que existem os valores a espaço que queremos preencher com a ajuda do CBR, começámos por atribuir os valores entre 0 e 1 às colunas restantes de forma a expressar a ordem de prioridades da lista mencionada anteriormente, ficando com os valores "normalizados".

```
weighting_factors = [
    0.02 %Age
    0.02 %Sex
    0.05 %ALB
    0.05 %ALP
    0.27 %ALT
    0.23 %AST
    0.20 %BIL
    0.04 %CHE
    0.03 %CHOL
    0.03 %CREA
    0.03 %GGT
    0.03 %PROT
];
```

*Figura 5 - Valores dados para o peso dos fatores*

Começamos por invocar a função *get_sex_similarities* que permite fazer ajustes na similaridade do sexo, invocar a função *get_max_values* que permite copiar o máximo de cada coluna, através da função *max* do Matlab, para um array *max_values, inicializar o array retrieved_indexes e similarities,* dar o valor de menos infinito à variável *best_similaritie* (valor que se pretende maximizar, por isso é que se dá o menor valor possível) e criar um *array list* com os nomes dos fatores que influenciam o diagnostico da hepatite.

```
function [sex_sim] = get_sex_similarities()

    sex_sim.categories = categorical({'m', 'f'});
    sex_sim.similarities = [
          1.0 0.8
          0.8 1.0
    ];
end
```

*Figura 6 - Função get_sex_similarities*

```
function [max_values] = get_max_values(case_library)
    key_set = {'Age','Sex','ALB','ALP','ALT','AST','BIL','CHE','CHOL','CREA','GGT','PROT'};
    value_set = {max(case_library{:,'Age'}), max(case_library{:,'Sex'}), ...
        max(case_library{:,'ALB'}), max(case_library{:,'ALP'}), max(case_library{:,'ALT'}), ...
        max(case_library{:,'AST'}), max(case_library{:,'BIL'}), max(case_library{:,'CHE'}), ...
        max(case_library{:,'CHOL'}), max(case_library{:,'CREA'}), max(case_library{:,'GGT'}), ...
        max(case_library{:,'PROT'})};
    max_values = containers.Map(key_set, value_set);
end
```

*Figura 7 - Função get_max_values*

De seguida, através de um ciclo *for* verifica-se cada um dos fatores presentes no novo caso, se uma característica não estiver preenchida no novo caso, então atribui-se um peso de 0 para esse fator (weighting_factors), garantindo que apenas os fatores relevantes estão presentes.

Seguidamente, através de outro ciclo *for* cria-se uma matriz com zeros (*distances*), para essa matriz vai o resultado do cálculo da distância linear de cada fator através da fórmula dada nas aulas práticas:

```
function [res] = calculate_linear_distance(val1, val2)
    res = sum(abs(val1-val2))/length(val1);
end
```

*Figura 8 - Fórmula da Distância Linear dada na aula*

```
if isfield(new_case, 'Age')
    distances(1,1) = calculate_linear_distance(case_library{i,'Age'} / max_values('Age'), ...
        new_case.Age / max_values('Age'));
end
```

*Figura 9 – Exemplo da aplicação da fórmula*

Depois é calculado a similaridade global aplicando a seguinte fórmula:

$$D_L(q,s) = \frac{\sum_{i=1}^{n} d(q_i, s_i) \times w_i}{\sum_{i=1}^{n} w_i}$$

| | |
|---|---|
| $D_L$ | Distância linear global entre casos |
| $D_E$ | Distância euclidiana global entre casos |
| $q$ | Caso novo |
| $s$ | Caso da biblioteca |
| $q_i$ | Valor do atributo de índice *i* no caso novo |
| $s_i$ | Valor do mesmo atributo no caso da biblioteca |
| $w_i$ | Fator de relevância (peso) para o atributo de índice *i* |
| $d$ | Função **distância local** entre atributos correspondentes |

*Figura 10 - Fórmula da similaridade global dada na aula*

```
final_similarity = 1 - sum(weighting_factors.*distances')/sum(weighting_factors);
```

*Figura 11 - Exemplo da aplicação da fórmula*

Esta similaridade é comparada com o limite mencionado anteriormente (*threshold*) para verificar se o valor é superior, se for superior é adicionado o índice do caso correspondente à matriz

*retrived_indexes* e o valor é copiado para a matriz *similarities*. Ainda no ciclo for mencionado anteriormente verifica-se se o valor obtido (*final_similarity*) é superior ao valor da melhor similaridade (*best_similaritie*), se sim dá-se o valor do *final_similarity à variável best_similaritie* e dá-se o caso que representa a melhor similaridade à varável *new_case*.

```
final_similarity = 1 - sum(weighting_factors.*distances')/sum(weighting_factors);

if final_similarity >= threshold
    retrieved_indexes = [retrieved_indexes i];
    similarities = [similarities final_similarity];
end

fprintf('Case %d de of %d tem semelhanca de %.2f%%...\n', i, size(case_library,1),

if final_similarity > best_similaritie && pos_new_case ~= i
    best_similaritie = final_similarity;
    new_case.Category = case_library{i, 'Category'};
end
```

*Figura 12 - Cálculo da similaridade final*

A distância global é calculada com base nas similaridades locais de cada atributo e nos pesos atributos a essas variáveis. A distância entre dois valores é calculada a média da diferença absoluta entre os campos, é útil quando os valores são numéricos e não há uma relação de ordem específica entre eles, logo é uma abordagem simples e eficaz para calcular a distância.

Por fim, segue um excerto a tabela de similaridade obtidas (pode encontrar o ficheiro *similarities*:

| ID | 4 | 121 | 344 | 360 | 562 | 578 | 582 | 602 |
|---|---|---|---|---|---|---|---|---|
| 1 | 96,220 | 96,343 | 94,966 | 94,670 | 86,156 | 93,820 | 65,446 | 78,314 |
| 2 | 96,232 | 95,346 | 92,154 | 93,209 | 90,095 | 90,460 | 68,188 | 77,714 |
| 3 | 97,601 | 97,381 | 94,961 | 94,830 | 87,525 | 91,177 | 67,716 | 77,374 |
| 4 | [redacted] | 97,678 | 95,362 | 95,228 | 88,431 | 92,038 | 67,175 | 76,793 |
| 5 | 96,459 | 97,276 | 95,326 | 95,036 | 85,754 | 91,906 | 65,557 | 77,655 |
| 6 | 96,400 | 97,265 | 96,183 | 95,485 | 86,192 | 92,375 | 65,406 | 77,068 |
| 7 | 97,124 | 96,008 | 94,024 | 93,531 | 87,996 | 92,144 | 69,388 | 77,646 |
| 8 | 97,576 | 98,099 | 95,833 | 95,538 | 86,581 | 91,532 | 65,357 | 76,154 |
| 9 | 95,223 | 95,035 | 94,670 | 93,910 | 84,709 | 91,504 | 68,257 | 80,209 |
| 10 | 96,048 | 96,340 | 94,899 | 94,548 | 86,406 | 92,344 | 67,590 | 77,592 |
| 11 | 96,461 | 96,833 | 95,871 | 95,360 | 85,869 | 93,243 | 64,738 | 77,887 |
| 12 | 98,067 | 97,989 | 95,825 | 96,521 | 86,861 | 92,815 | 65,945 | 77,812 |

*Figura 13 - Tabela de similaridade*

Ainda sobre a função TP, dentro do ciclo que chama o *retrieve*, são copiados para uma nova variável *retrieved_cases* o caso com maior semelhança (resultado do CBR) e a sua similaridade (*retrieved_cases.Similarities*), por fim, substitui-se a categoria com um espaço vazio com a categoria com maior similaridade. No fim, escreve-se num ficheiro (*retrieved_filled.csv*) a variável *case_library* agora com os espaços completados, ou seja, a categoria neste ficheiro encontra-se sem NA.

```
            retrieved_cases = case_library(retrieved_indexes, :);
            retrieved_cases.Similarity = similarities';

            case_library.Category(i) = new_case.Category;
        end
    end

    fprintf('\nFase de Retrieve concluida\n');

    writetable(case_library,'retrieved_filled.csv');
```

*Figura 14 - Fim da função TP*

# Redes Neuronais

As redes neurais são modelos computacionais inspirados no funcionamento do cérebro humano. Compostas por neurónios artificiais interligados, organizam-se em camadas e têm a capacidade de aprender a partir de dados, identificando padrões complexos e executando tarefas de classificação, regressão, reconhecimento de padrões, entre outras. Cada neurónio recebe entradas, realiza uma operação de ponderação e soma, aplica uma função de ativação e produz uma saída. As redes neurais organizam-se em camadas, incluindo uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Durante o treino, as redes neurais ajustam os pesos das conexões entre os neurónios para minimizar a diferença entre as saídas esperadas e as saídas reais, utilizando algoritmos de otimização como o gradiente descendente. As redes neurais têm a capacidade de aprender a partir de exemplos e generalizar para novos dados.

No trabalho proposto envolve a aplicação de redes neurais *feedforward* para "diagnosticar" a hepatite. As redes neurais são treinadas e analisadas utilizando a *toolbox Deep Learning* do Matlab. Os objetivos incluem o estudo da influência das diferentes arquiteturas de redes, funções de treino e de ativação, bem como a análise da capacidade de generalização das redes treinadas. As redes neurais são utilizadas em conjunto com o CBR para preencher valores ausentes e realizar tarefas de classificação com base em casos semelhantes previamente observados.

# One Hot encoding

Na função *Start_Redes* começámos por ler o ficheiro Excel (*Start*) para uma matriz *data* e separamos os dados em *inputs* (todas as colunas, excluindo o ID e a *Category*) e targets (*Category)*.

```
inputs = (data(:, 3:end)');
targets = data(:, 2)';
```

*Figura 15 - Divisão dos dados*

Como dados são *categorical* tem de se transformar cada categoria num vetor binário (zeros e uns), uma vez que os algoritmos das redes neuronais operam com variáveis numéricas (ou seja, para ser compatível com o algoritmo e evitar interpretações de ordem dos dados).

Para transformar os dados, define-se o número de classes que o modelo deve distinguir, neste caso são 5 (*numClasses*), posteriormente, inicializa-se uma matriz de zeros com o tamanho dos dados para armazenar os dados transformados pelo *one-hot coding (oneHotTargets).* Depois itera-se os dados do *target* de forma a dar-se então o vetor binário correspondente (o vetor tem 5 posições em que cada posição representa um atributo da categoria; A posição zero é o *Blood Donor*, a posição um é *suspect Blood Donor*, a posição dois é Hepatite, a posição três é a Fibrose e a posição quatro é Cirrose, ou seja, se aquele caso for *Blood Donor* ficará com o vetor [1 0 0 0 0]).

```
numClasses = 5;
oneHotTargets = zeros(numClasses, length(targets));

for i = 1:length(targets)
    switch targets(i)
        %tipos de doadores
        case 0
            oneHotTargets(:, i) = [1 0 0 0 0]';
        case 1
            oneHotTargets(:, i) = [0 1 0 0 0]';
        case 2
            oneHotTargets(:, i) = [0 0 1 0 0]';
        case 3
            oneHotTargets(:, i) = [0 0 0 1 0]';
        case 4
            oneHotTargets(:, i) = [0 0 0 0 1]';
    end
end
```

*Figura 16 - One-Hot Coding*

Agora com as variáveis prontas passa-se para a criação e aplicação das redes neuronais.

# Feedfoward net

A *feedfoward net* é uma rede neuronal em que a informação flui numa única direção (da entrada para a saída) sem formar ciclos. Simplificando, é um tipo de arquitetura de uma rede neuronal em que os dados são passados diretamente da camada de entrada para a camada de saída, sem "*feedback*" das previsões anteriores ou conexões retroativas ("*que voltam para trás*").

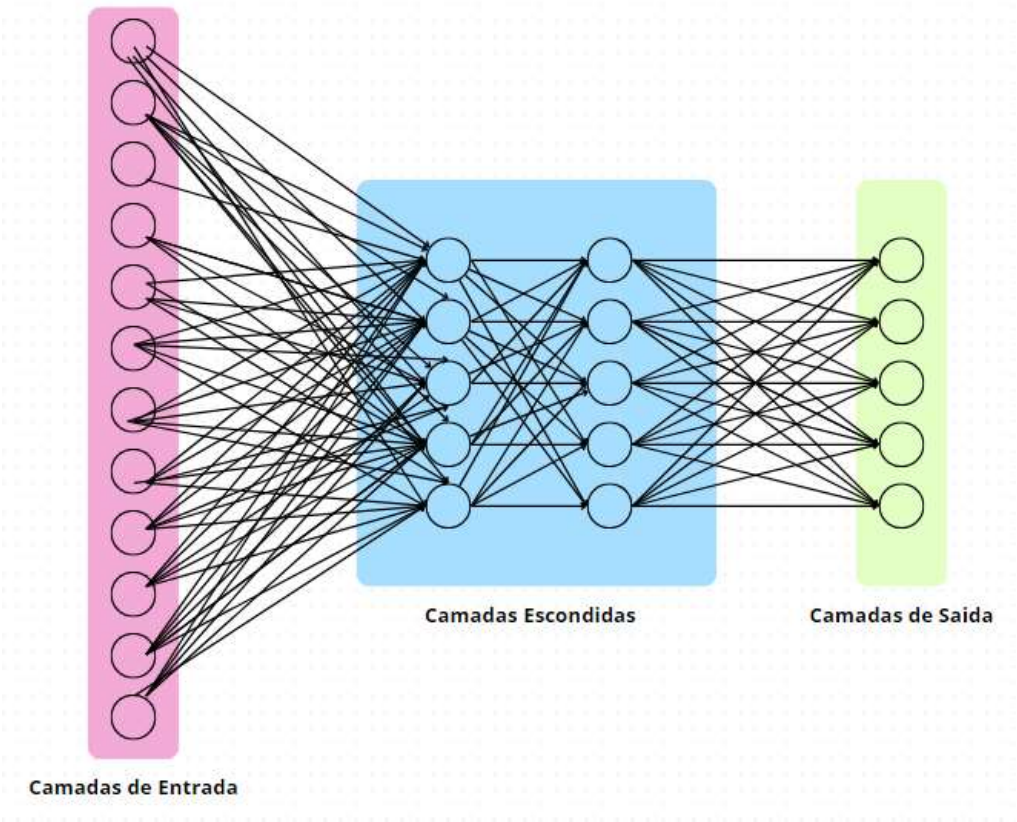



*Figura 17 - Arquitetura da rede neuronal do nosso trabalho*

A nossa arquitetura representada pela imagem acima é constituída por doze pontos de entrada (inputs) que representa os fatores de decisão para o diagnóstico (ou seja, a idade, o sexo, ALP, ...), uma camada escondida constituída por dez neurónios e cinco pontos de saída que representa os atributos da categoria (outputs).

Ainda no ficheiro *Start_redes* cria-se a rede neuronal com dez neurónios, configura-se a rede com as funções de treino, ativação e saída e divide-se os dados em dados para treino, dados para validação e dados para teste através de uma percentagem (preparação da rede neuronal para treino).

```
net = feedforwardnet(10);
%Função de treino
net.trainFcn = 'trainlm';
%Função de Ativação
netlayers{1}.transferFcn = 'tansig';
%Função de Saida
netlayers{2}.transferFcn = 'purelin';
%Dividir os dados
net.divideFcn = 'dividetrain';
net.divideParam.trainRatio = 0.7; %dados para treino
net.divideParam.valRatio = 0.15; %dados para validação
net.divideParam.testRatio = 0.15; %dados para teste
```

*Figura 18 - Configuração da rede neuronal*

Por fim, treina se a rede e faz-se a sua simulação, com isto obtemos o erro e a precisão, com isto acaba o programa do *Star_redes*.

```
%Treino da rede
net = train(net, inputs, oneHotTargets);

%Simulação da rede
out = sim(net, inputs);

%Cálculo do Erro
erro = perform(net,out,oneHotTargets);
accuracy = (1-erro) * 100;
```

*Figura 19 - Treino e Simulação da rede neuronal*

## *Análise de Dados do Start*

Após o desenvolvimento da função explicada, procedemos à realização de vários testes [7]de forma a ser efetuada uma análise que mostre quais os parâmetros que mais afetam o treino e a simulação da rede neuronal.

Em primeiro ligar, definimos os seguintes valores de forma a decidir uma configuração *default*:

- Número de Testes: 30;
- Número de Camadas Escondidas: 1;
- Número de Neurónios: 10;
- Funções de Ativação: *tansig*, *purelin*;
- Função de treino: *trainlm*;
- Divisão para treinar, validar e testar: 0.7, 0.15, 0.15

O resultado da configuração *default* foram a seguinte:

| Tempo de Execução | Erro na Classificação | Precisão Global |
|---|---|---|
| 0,63 | 0 | 100 |

*Figura 20 - Resultado da configuração default (start)*

De seguida, foram realizados vários testes, onde foram mudadas as **funções de treino**, foram usadas as seguintes funções: *traingd*, *trainbfg*, *traingdx*, *trainscg*, *trainbfg*, ao realizar estas mudanças ocorreu um **elevado aumento do tempo de execução**, no entanto, **o erro de classificação manteve se igual**. Quando a função de *traingd* é usada o erro de classificação aumenta ligeiramente e consequentemente a precisão global sofre uma descida.

[7] **Nota:** Os dados relativos aos testes está no ficheiro Excel **Resultados**.

| | | | | |
|---|---|---|---|---|
| traingd | dividerand = [0.7, 0.15, 0.15] | 1,31 | 0,1 | 90,47 |
| trainbfg | dividerand = [0.7, 0.15, 0.15] | 2,3 | 0 | 100 |
| traingdx | dividerand = [0.7, 0.15, 0.15] | 1,17 | 0 | 100 |
| trainscg | dividerand = [0.7, 0.15, 0.15] | 0,67 | 0 | 100 |
| trainbfg | dividerand = [0.7, 0.15, 0.15] | 0,67 | 0 | 100 |

*Figura 21 - Resultado das diferentes configurações da função de treino (start)*

Foram realizados mais testes, neste caso variou se as **funções de ativação**, as combinações realizadas foram as seguintes *losig* & *purelin*, *tansig & logsig, purelin & softmax, logsig & softmax, softmax & softmax, tansig & softmax*. Com estas combinações deparamo-nos com um **aumento do tempo de execução**, maior em relação ao referido anteriormente, mas o **erro de classificação foi sempre 0**, logo a **precisão global foi sempre 100**.

| | | | | | |
|---|---|---|---|---|---|
| logsig, purelin | trainlm | dividerand = [0.7, 0.15, 0.15] | 1,57 | 0 | 100 |
| tansig, logsig | trainlm | dividerand = [0.7, 0.15, 0.15] | 2,12 | 0 | 100 |
| purelin,softmax | trainlm | dividerand = [0.7, 0.15, 0.15] | 0,69 | 0 | 100 |
| logsig,softmax | trainlm | dividerand = [0.7, 0.15, 0.15] | 3,66 | 0 | 100 |
| softmax,softmax | trainlm | dividerand = [0.7, 0.15, 0.15] | 2,66 | 0 | 100 |
| tansig, softmax | trainlm | dividerand = [0.7, 0.15, 0.15] | 1,95 | 0 | 100 |

*Figura 22 - Resultado das diferentes configurações da função de ativação (start)*

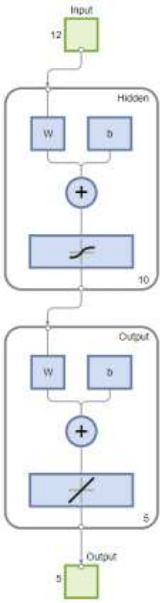



*Figura 23 - Arquitetura da rede com melhor configuração (start)*


*Figura 24 - Matriz de confusão da melhor configuração (start)*

Em suma, podemos concluir que a alteração dos parâmetros: função de ativação e função de treino, não causa um grande impacto

nos resultados da rede, apenas aumenta o tempo de execução. Com isto, conseguimos verificar que a rede está bem ajustada para os dados deste ficheiro, porque é capaz de aprender completamente os padrões nos dados de treino, verificando também que as melhores configurações para estes dados são as que a precisão alcança o mais alto valor.

### *Função treino_redes*

Esta função recebe as configurações da rede por argumento (o número de neurónios, a função de treino, a função de ativação, a função de saída e a divisão de dados para treino, validação e teste) e devolve precisão global, o tempo de execução e o erro global.

À semelhança da função *Start_redes* começa-se por ler o ficheiro Excel preenchido anteriormente com a ajuda do CBR (*retrieved_filled*), separa-se os dados (como se fez na função *Start_redes*) e aplica-se mais uma vez o método *oneHotTargets*.

Inicializa-se as matrizes para guardar a precisão total (*accuracy_total*) e a precisão teste (*accuracy_teste*), de seguida, cria-se a rede com a configuração inserida nos parâmetros de entrada da função, treina-se a rede e simula-se (e guarda se o erro de precisão da rede).

Seguidamente, guarda-se os valores mais alto da saída obtida e da saída desejada e se estes forem iguais a classificação da rede neuronal teve sucesso (incrementa -se o *r* quando isso acontece), com estes valores calcula-se a precisão do total.

```
r=0;
for i=1:size(out,2)
  [a b] = max(out(:,i));
  [c d] = max(oneHotTargets(:,i));
  if b == d
      r = r+1;
  end
end
```

*Figura 25 - Análise do sucesso da rede*

```
accuracyTotal = r/size(out,2)*100;
accuracy_total=[accuracy_total accuracyTotal];
```

*Figura 26 - Cálculo da precisão total*

De seguida, separa-se os dados para a parte do teste e simula-se a rede, repetindo o processo mencionado acima para guardar os valores mais altos.

```
TInput = inputs(:,tr.testInd);
TTargets = oneHotTargets(:,tr.testInd);
out = sim(net, TInput);
erroT = perform(net, out,TTargets);

r=0;
for i=1:size(tr.testInd,2)
  [a b] = max(out(:,i));
  [c d] = max(TTargets(:,i));
  if b == d
      r = r+1;
  end
end
```

*Figura 27 - Separa os dados e simular para o teste*

Ao contrário do que foi feito anteriormente transforma-se a matriz de saída da rede neuronal (*out*) numa matriz binária, transformando as percentagens acima de 50% em 1 e as inferiores ou

iguais em 0, de seguida, calcula-se o número de atributos da *category* corretamente classificadas através da comparação com os atributos da *category* presentes no ficheiro. Calcula-se a precisão de teste e guarda-se esse valor e o tempo de execução.

```
out = (out>0.5); %transformar em 0 ou 1
r = sum(out == TTargets,2);
accuracyTeste = r/size(tr.testInd,2)*100;
accuracy_teste = [accuracy_teste accuracyTeste];


tempo_execucao = toc;
tempo_execucao = [tempo_execucao tempo_execucao];
```

*Figura 28 - Transformação do out e calculo da precisão do teste*

Por fim, calcula-se a média da precisão do teste (por outras palavras, precisão do teste global) e a média do tempo de execução. Para guardar as três melhores redes usa-se a precisão total e inicializa-se uma matriz com o primeiro valor da precisão total e com dois valor mais negativo (menos infinito), de seguida, é criado um ciclo para compara as precisões globais entre si mesmas, guardando as que tiverem maior valor. Por fim, guarda-se as três redes neuronais resultantes com o *save*.

```
accuracyI = accuracy_total(1);
accuracyThree = [accuracyI -Inf -Inf];
for i=2:length(accuracy_total)
    if (accuracyI >= accuracy_total(i))
        for j=1:3
            if (accuracyI >= accuracyThree(j))
                accuracyThree(j) = accuracyI;
            end
        end
    end
end


%save('treinoRede_config1.mat','net');
```

*Figura 29 -Cciclo para descobrir as três melhores redes neuronais*

## *Analise de Dados do Treino*

Após o desenvolvimento da função acima explicada, procedemos à realização de vários testes de forma a ser efetuada uma análise que mostre quais os parâmetros que mais afetam o treino e simulação da rede neuronal.

Primeiro corremos o programa com os seguintes valores de forma a definir uma configuração *default*:

- Número de Testes: 30;
- Número de Camadas Escondidas: 1;
- Número de Neurónios: 10;
- Funções de Ativação: *tansig*, *purelin*;
- Função de treino: *trainlm*;
- Divisão para treinar, validar e testar: 0.7, 0.15, 0.15

O resultado da configuração *default* foram a seguinte:

| Tempo de Execução | Erro na Classificação | Erro na Classificação do Teste | Precisão Global | Accuracy Teste |
|---|---|---|---|---|
| 0,77 | 0,01 | 0,02 | 95,27 | 97,02 |

*Figura 30 - Resultado da configuração default (treino)*

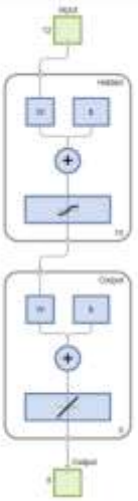


*Figura 31 - Arquitetura da rede com a configuração default(treino)*

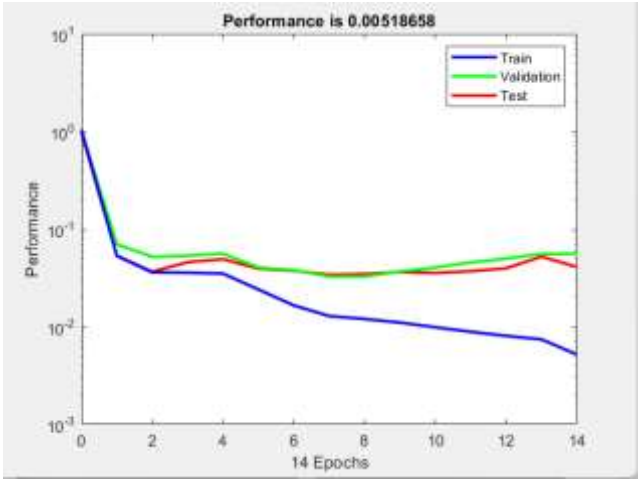



*Figura 32 - Gráfico da Perfomance da Rede com a configuração default (treino)*


*Figura 33 - Matriz de Confusão da Rede com a configuração default (treino)*

Começamos por efetuar testes de forma a verificar o **impacto do número de neurónios e camadas escondidas na rede neuronal**. Após vários testes chegamos a conclusão de que a **existência de duas camadas escondidas com o número de neurónios iguais**,

obtém os **melhores resultados** com o **melhor tempo de execução**, erro **de classificação global, o erro de classificação de teste, precisão global e precisão de teste.**

| | | | | | | |
|---|---|---|---|---|---|---|
| 2 | 5 , 5 | 0,88 | 0,01 | 0,02 | 94,8 | 97,13 |
| 2 | 10 , 10 | 0,9 | 0,02 | 0,04 | 95,68 | 97,04 |
| 3 | 5 , 10 , 5 | 0,96 | 0,02 | 0,03 | 94,8 | 96,82 |
| 4 | 10 , 5 , 10 , 5 | 1,07 | 0,01 | 0,02 | 95,24 | 96,62 |

*Figura 34 -Resultado dos diferentes números de neurónios e de camadas (treino)*

De seguida, variamos as **funções de treino**, foram usadas as seguintes funções: *traingd*, *trainbfg*, *traingdx*, *trainscg*, *trainbfg*. Concluímos as melhores funções de treino foram a ***trainbfg*** e a ***trainscg***, mas em geral o **tempo de execução, o erro de classificação global, o erro de classificação do teste e a precisão global e a precisão de teste**, **piorou** em relação às experiências anteriores.

| | | | | | | |
|---|---|---|---|---|---|---|
| traingd | dividerand = (0.7, 0.15, 0.15) | 1,6 | 0,06 | 0,05 | 87,8 | 94,8 |
| trainbfg | dividerand = (0.7, 0.15, 0.15) | 1,01 | 0,02 | 0,03 | 91,57 | 96,68 |
| traingdx | dividerand = (0.7, 0.15, 0.15) | 0,92 | 0,03 | 0,04 | 91,4 | 96,56 |
| trainscg | dividerand = (0.7, 0.15, 0.15) | 0,72 | 0,03 | 0,03 | 91,57 | 96,26 |
| trainbfg | dividerand = (0.7, 0.15, 0.15) | 1 | 0,03 | 0,04 | 91,44 | 96,69 |

*Figura 35 - Resultado das diferentes configurações da função de treino (treino)*

Neste caso variou se **combinações das funções de ativação**, realizaram-se as seguintes combinações*: losig & purelin, tansig & logsig, purelin & softmax, logsig & softmax, softmax & softmax, tansig & softmax*. Concluímos que existe uma **grande gama de valores**, deparamo-nos com **bons e maus valores na precisão global** e na **precisão de teste**, uma **diminuição do tempo de execução** e o **erro teve um grande aumento** nos casos em que a precisão de teste se revela pequena.

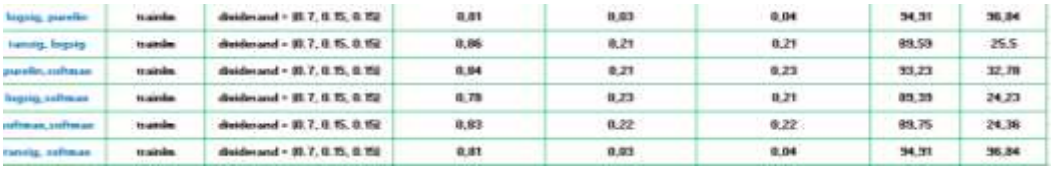


*Figura 36 - Resultado das diferentes configurações da função de ativação (treino)*

De seguida, foram criadas **várias combinações para as divisões dos pesos de Treino de Validação e de Teste**. Encontramos **valores muito positivos em relação ao tempo de execução ao erro de classificação global, ao erro de classificação global teste, a precisão global e a precisão de teste**.

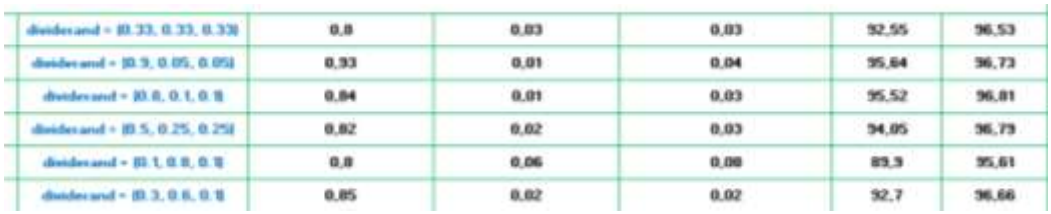


*Figura 37 - Resultado das diferentes configurações da divisão dos dados(treino)*

Em suma, as três melhores configurações são as seguintes:

```
net = feedforwardnet([5 5]);
net.trainFcn = 'trainlm';
net.layers{1:end-1}.transferFcn = 'tansig';
net.layers{end}.transferFcn = 'purelin';
net.divideFcn = 'dividerand';
net.divideParam.trainRatio = 0.7;
net.divideParam.valRatio = 0.15;
net.divideParam.testRatio = 0.15;
```

```
net = feedforwardnet([10 10]);
net.trainFcn = 'trainlm';
net.layers{1:end-1}.transferFcn = 'tansig';
net.layers{end}.transferFcn = 'purelin';
net.divideFcn = 'dividerand';
net.divideParam.trainRatio = 0.7;
net.divideParam.valRatio = 0.15;
net.divideParam.testRatio = 0.15;
```

```
net = feedforwardnet;
net.trainFcn = 'trainlm';
net.layers{1:end-1}.transferFcn = 'logsig';
net.layers{end}.transferFcn = 'purelin';
net.divideFcn = 'dividerand';
net.divideParam.trainRatio = 0.7;
net.divideParam.valRatio = 0.15;
net.divideParam.testRatio = 0.15;
```

*Figura 38 – Três melhores configurações da rede (treino)*

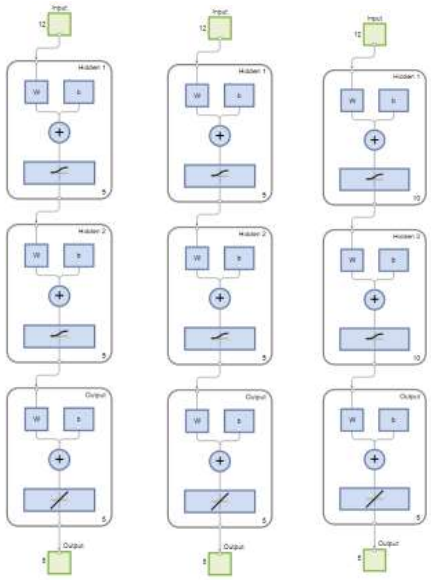


*Figura 39 - Arquitetura das três melhores redes (treino)*

Concluindo, a alteração dos parâmetros acima referidos irá causar um grande impacto na rede, ao nível de todos os resultados, também conseguimos verificar que a rede está bem ajustada para os dados do ficheiro, porque é capaz de aprender completamente os padrões nos dados de treino, verificando também que as melhores configurações para estes dados são as que a precisão alcança o mais alto valor, e a divisão de dados é a indicada.

## *Função Teste_redes*

Esta função recebe o nome da rede neuronal e os dados que são os valores das colunas por argumento e devolve precisão global, o tempo de execução, o erro global e a categoria.

À semelhança da função *treino_redes* começa-se por ler o ficheiro Excel (*Test*), carrega uma rede neuronal anteriormente treinada, separa-se os dados (como se fez na função *Start_redes*) e aplica-se mais uma vez o método *oneHotTargets*.

Como se fez para o *treino*_redes, inicializa-se as matrizes para guardar a precisão total (*accuracy_total*) e a precisão teste

(*accuracy_teste*) e simula-se a rede (e guarda se o erro de precisão da rede). Seguidamente, guarda-se os valores mais alto da saída obtida e da saída desejada e se estes forem iguais a classificação da rede neuronal teve sucesso (incrementa -se o r quando isso acontece), com estes valores calcula-se a precisão do total. Transforma-se a matriz de saída da rede neuronal (*out*) numa matriz binária, transformando as percentagens acima de 50% em 1 e as inferiores ou iguais em 0, de seguida, calcula-se o número de atributos da *category* corretamente classificadas através da comparação com os atributos da *category* presentes no ficheiro. Calcula-se a precisão de teste e guarda-se esse valor e o tempo de execução. Calcula-se as médias do tempo de execução e da precisão global.

Ao contrário do que fizemos na função anterior, esta alínea é adaptada para o posterior uso da aplicação, uma vez que calcula a categoria e mostra-a na *label* da aplicação e posteriormente, na linha de comandos.

```
if ~isempty(dados)
    fprintf('A avaliar...\n');
    [v, Cate] = max(out);
    Cate = Cate - 1;
    fprintf('Categoria prevista para o exemplo: %d, valor da rede: %.2f\n', Cate, v);
else
    Cate = zeros(1, size(out, 2));
    for i = 1:size(out, 2)
        [v, Cate] = max(out(:, i));
        [~, f] = max(oneHotTargets(:, i));
        Cate = Cate - 1;
        f = f - 1;
        fprintf('Categoria prevista para o exemplo %d: %d, valor da rede: %.2f, valor desejado: %d\n', i, Cate, v, f);
        Cate(i) = Cate;
    end
```

*Figura 40 - Código para mostrar a categoria na aplicação*

## *Analise de Dados do Teste*

Relativamente às saídas das redes com os *targets* desejados, a rede com melhor configuração obtém as melhores métricas de acerto com uma **percentagem de 86%**. Para comprovar este resultado foram testados valores referentes aos vários valores da *Category*:


*Figura 41 - Identificação do caso Blood Donor*


*Figura 42- Identificação do caso suspect Blood Donor*


*Figura 43 - Identificação do caso Hepatitis*


*Figura 44 - Identificação do caso Fibrosis*


*Figura 45 - Identificação do caso Cirrhosis*

Com estas imagens concluímos que a rede com melhor configuração, revela o target desejado, em todos os casos, no entanto, por vezes falha o diagnóstico da Hepatite.

# Aplicação em Matlab

Para tornar o código mais interativo criamos uma aplicação gráfica do treino e do teste, por outras palavras, uma aplicação que permite criar uma rede neuronal, configurar a rede, treinar a rede, carregar uma rede à escolha e modificar os valores dos atributos na parte do teste.

## *Treino*

Como pode observar na imagem que se segue do lado esquerdo tem os dados, onde pode alterar a configuração da rede neuronal e do lado direito tem os resultados e o botão treino.

Para utilizar basta então inserir valores válidos nos dados (se não forem válidos aparece um erro), clicar no botão treino e depois observar os resultados. Nota: enquanto está a ocorrer o treino o botão fica a vermelho, por favor aguarde até este voltar à cor original.


*Figura 46 - Painel inicial da app (treino)*


*Figura 47 - Aplicação em execução (treino)*


*Figura 48 - Exemplo do resultado (treino)*

## *Teste*

Como pode observar na imagem que se segue do lado esquerdo tem os dados, onde pode alterar os valores das diferentes colunas que estavam presentes no Excel e a opção de selecionar uma rede neuronal treinada anteriormente. Do lado direito tem os resultados, o botão teste, o botão resultado e o resultado da categoria.

Para utilizar basta então inserir valores válidos nos dados (se não forem válidos aparece um erro), inserir o nome da rede neuronal, clicar no botão teste e depois observar os resultados, depois clicar no botão resultado para observar a categoria.


*Figura 49 - Painel inicial da app (teste)*


*Figura 50 - Exemplo do resultado (teste)*

## *Info*

Nesta parte poderá observar os nossos nomes e número de alunos do lado direito e do lado esquerdo tem dois botões um para abrir o relatório e outro para sair da aplicação.


*Figura 51 - Painel inicial da app (info)*

# Conclusão

O objetivo final é não apenas compreender os princípios teóricos subjacentes ao CBR e redes neurais, mas também aplicar esses conceitos de forma prática num contexto relevante, como o diagnóstico médico. Através deste trabalho, desenvolvemos competências em análise de dados, preparação de conjuntos de dados, treino e avaliação de redes neurais, e comunicação de resultados através de relatórios técnicos.

# Referências Bibliográficas

Hepatite - o que é, causas e tipos, sintomas, tratamento, cura. (n.d.). Www.saudebemestar.pt. https://www.saudebemestar.pt/pt/clinica/gastrenterologia/hepatite

*Hepatite: o que é, sintomas, causas e tratamento*. (n.d.). Tua Saúde. https://www.tuasaude.com/hepatite/

*Hepatite*. (2022, May 1). Wikipedia. https://pt.wikipedia.org/wiki/Hepatite

*Vírus da hepatite. A (VHA)*. (n.d.). SNS24. https://www.sns24.gov.pt/tema/doencas-infecciosas/virus-da-hepatite-a-vha/

Kumar, S. (2022, August 2). *Considerações gerais sobre a hepatite*. Manual MSD Versão Saúde Para a Família; Manuais MSD. https://www.msdmanuals.com/pt-pt/casa/doen%C3%A7as-hep%C3%A1ticas-e-da-ves%C3%ADcula-biliar/hepatite/considera%C3%A7%C3%B5es-gerais-sobre-a-hepatite

Kumar, S. (2022, August 2). *Causas da hepatite*. Manuais MSD Edição Para Profissionais; Manuais MSD. https://www.msdmanuals.com/pt-pt/profissional/dist%C3%BArbios-hep%C3%A1ticos-e-biliares/hepatite/causas-da-hepatite

Vírus da hepatite B (VHB. (2019, May 2). *Centro de Contacto Serviço Nacional de Saúde*. SNS24. https://www.sns24.gov.pt/tema/doencas-infecciosas/vhb/

Kumar, S. (2022, August 2). *Considerações gerais sobre a hepatite*. Manual MSD Versão Saúde Para a Família; Manuais MSD. https://www.msdmanuals.com/pt-pt/casa/doen%C3%A7as-hep%C3%A1ticas-e-da-ves%C3%ADcula-biliar/hepatite/considera%C3%A7%C3%B5es-gerais-sobre-a-hepatite

Kumar, S. (2022, August 2). *Hepatite A*. Manual MSD Versão Saúde Para a Família; Manuais MSD. https://www.msdmanuals.com/pt-pt/casa/doen%C3%A7as-hep%C3%A1ticas-e-da-ves%C3%ADcula-biliar/hepatite/hepatite-a-aguda

Kumar, S. (2022, August 2). *Hepatite A*. Manuais MSD Edição Para Profissionais; Manuais MSD. https://www.msdmanuals.com/pt-pt/profissional/dist%C3%BArbios-hep%C3%A1ticos-e-biliares/hepatite/hepatite-a

hepato. (n.d.). *Interpretando os testes diagnósticos nas hepatites A, B e C | Hepato*. https://hepato.com/2009/04/20/interpretando-os-testes-diagnosticos-nas-hepatites-a-b-e-c-2/

Kumar, S. (2022, August 2). *Considerações gerais sobre a hepatite viral aguda*. Manual MSD Versão Saúde Para a Família; Manuais MSD. https://www.msdmanuals.com/pt-pt/casa/doen%C3%A7as-hep%C3%A1ticas-e-da-ves%C3%ADcula-biliar/hepatite/considera%C3%A7%C3%B5es-gerais-sobre-a-hepatite-viral-aguda Pinheiro, D. P. (2010, March 3). *Hepatite A: transmissão, sintomas, vacina e tratamento*. Www.mdsaude.com. https://www.mdsaude.com/gastroenterologia/hepatite-a/

Camacho, V. R. R., Silveira, T. R. da, Oliveira, J. R. de, Barros, S. G. S. de, & Cerski, C. T. S. (2007). Relação entre concentrações séricas de procolágeno tipo III, ácido hialurônico com achados histopatológicos do fígado em doadores de sangue anti-HCV positivos. *Arquivos de Gastroenterologia*, *44*, 118–122. https://doi.org/10.1590/S0004-28032007000200006

Garcia, F. B. (2024). Avaliação sorológica e epidemiológica para hepatite C dos doadores de sangue do Hemocentro Regional de Uberaba. *Uftm.edu.br*. http://localhost:8080/tede/handle/tede/26

Ferreira, O. (2007, April 27). *Estudo de doadores de sangue com sorologia reagente para hepatites B e C, HIV e sífilis no Hemocentro de Ribeirão Preto*. Www.teses.usp.br. https://www.teses.usp.br/teses/disponiveis/17/17139/tde-18032008-140000/en.php

Valente, V. B., Covas, D. T., & Passos, A. D. C. (2005). Marcadores sorológicos das hepatites B e C em doadores de sangue do Hemocentro de Ribeirão Preto, SP. *Revista Da Sociedade Brasileira de Medicina Tropical*, *38*(6), 488–492. https://doi.org/10.1590/s0037-86822005000600008