



Departamento de Engenharia Informática e Sistemas

Licenciatura em Engenharia Informática

Unidade Curricular Conhecimento e Raciocínio

Tema 1 - Redes Neurais

Relatório

Bernardo Godinho Vitória, 21250452
Filipe Ramos Alves, 21240035

Ano Lectivo 2017/2018

Índice

3.1 Estratégia Inicial	7
3.2 Estratégia Implementada	7
4.1 Inputs	8
4.1.1 Rotação	9
4.1.2 Redimensionamento	9
4.1.3 Conversão para Dados	10
4.1.3.1 Extração de pontos	10
4.1.3.2 Matriz Binária	10
4.2 Tratamento dos Targets	11
5.1 Funções de treino	12
5.2 Funções de ativação	12
5.3 Topologias	12
6.1- Implementação de uma rede neuronal para classificar por espécies	13
6.1.1 Feedforwardnet	13
Funções de treino	13
Função de ativação	14
6.1.2 Patternnet	16
Funções de treino	16
6.1.3 Cascadeforwardnet	17
Funções de treino	17
6.1.4 Fitnet	18
Funções de treino	18
Conclusão	19
6.2 Implementação de uma rede neuronal para classificar por subespécies.	20
6.2.1 Feedforwardnet	20
Função de treino	20
Função de ativação	21
6.2.2 Patternnet	22
Função de treino	22
6.2.3 Cascadeforwardnet	23
Função de treino	23
6.2.4 Fitnet	24
Função de treino	24
Conclusão	25
7.1 Feedforwardnet	26

Funções de treino	26
Funções de ativação	27
Matrix Confusão	28
Testes	28
7.2 Fitnet	29
Funções de treino	29
Matrix Confusão	30
Testes	30
7.3 Patternnet	31
Funções de treino	31
Funções de ativação	32
Matrix Confusão	33
Testes	33
7.4 Cascadeforwardnet	34
Funções de treino	34
Funções de ativação	35
Matrix Confusão	36
Figura 10 -“Matriz de confusão cascadeforwardnet”	36
Testes	36
Conclusão	37
Treinar redes neuronais	42
Configurar a topologia das redes neuronais	42
Configurar Hidden Layers/Neurônios	42
Escolher funções de treino / ativação	42
Gravar uma rede neuronal treinada	42
Figura 13 -“Painel de configuração do treino das redes”	42
Carregar uma rede neuronal	43
Visualizar os resultados da classificação	43
Geração/gravação dos resultados assim como as configurações da rede usada	43
Aplicar uma rede neuronal para classificar folhas	44
Figura 15 -“Classificar Imagem”	44
DATASET	45
problema	45
solução	45
EXERCÍCIO A (feito com extração de dados)	46
12.1 Feedforwardnet	46
Funções de treino	46
Funções de ativação	46
12.2 Fitnet	47

Funções de treino	47
Funções de ativação	47
12.3 Patternnet	48
Funções de treino	48
Funções de ativação	48
12.4 Cascadeforwardnet	49
Funções de treino	49
Funções de ativação	49
Conclusão	50

1- Introdução

O seguinte relatório diz respeito ao trabalho realizado sobre o tema “Redes Neurais”, no âmbito da unidade curricular de Conhecimento e Raciocínio, do segundo ano. O objetivo do trabalho consiste na implementação de uma rede neuronal capaz de classificar corretamente um conjunto de folhas pertencentes a cerca de 35 espécies diferente, que depois se subdividem em diferentes subespécies (99), o mesmo problema foi implementado por MATLAB.

No decorrer deste trabalho, iremos testar diversas topologias, diversas funções de avaliação assim como diferentes tratamentos de dados, tanto de Input como Output com o objectivo de otimizar a Rede Neuronal e adquirir mais conhecimentos de como estas se comportam e como podem ser melhoradas.

2- Redes Neurais

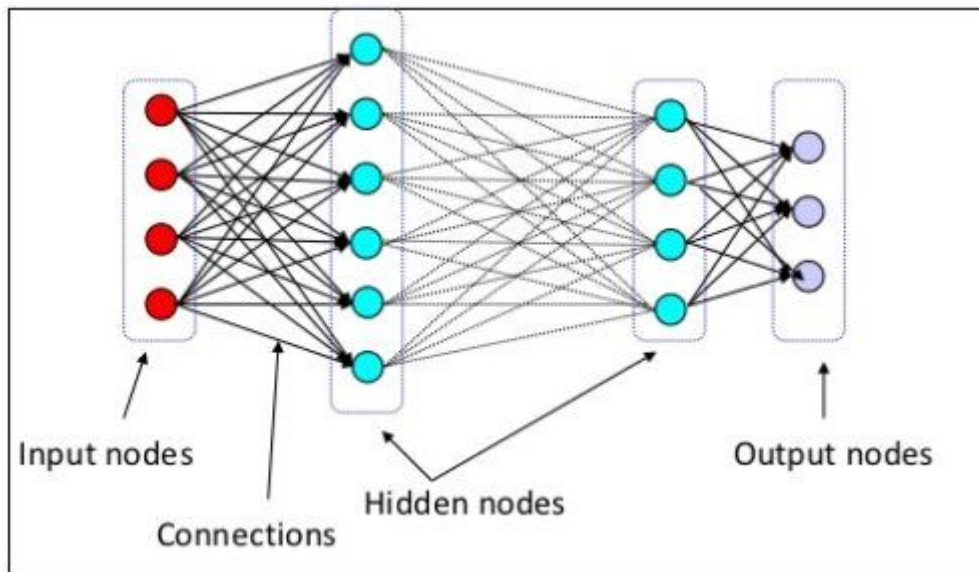


Figura 1 - “Representação de gráfica de uma Rede Neuronal”

As redes neurais constituem um suporte geral e prático para aprendizagem de modelos de classificação.

Uma Rede Neuronal é composta por unidades, cada unidade simula o funcionamento de um neurônio, essas mesmas unidades têm o nome de percepção.

O conceito de Rede Neuronal provém da analogia biológica, cérebro que equivale neurônios densamente interligados, cada neurônio recebe inputs de outros neurônios e produz uma saída que se liga a outros neurônios.

No âmbito do nosso trabalho prático podemos considerar o input como a imagem que representa a folha e o output como a classificação dela.

3- Estratégia de Implementação

3.1 Estratégia Inicial

Inicialmente a estratégia tinha como objectivo ter duas redes neuronais, a primeira rede neuronal iria determinar a Espécie, a segunda rede neuronal iria usar tanto a imagem como a informação obtida pela primeira rede neuronal de forma a rede neuronal conseguir reduzir a quantidade de respostas possíveis (para cada SubEspécie só existe uma única espécie).

3.2 Estratégia Implementada

Após implementada a estratégia inicial, reparámos que todo este processamento adicional não estava a demonstrar resultados muito melhores do que a uma rede a determinar directamente a Subespécie a partir da imagem, no então apercebemo-nos que a imagem apenas era o principal influenciador devido ao peso de “input’s”, decidimos então optar por uma abordagem, como ao exemplo da Íris lecionado nas aulas, e em vez de obtermos a imagem, iremos extrair os dados mais relevantes.

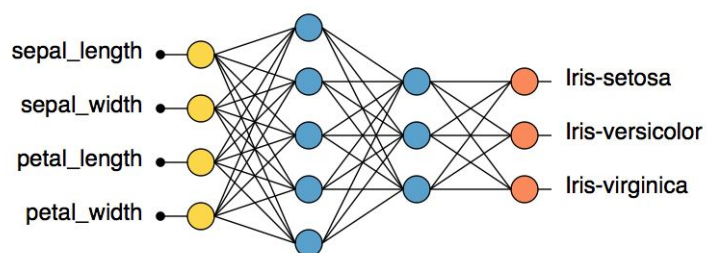


Figura 2 - “Representação de gráfica da Rede Neuronal lecionada nas aulas”

4- Implementação das Redes Neurais

4.1 Inputs

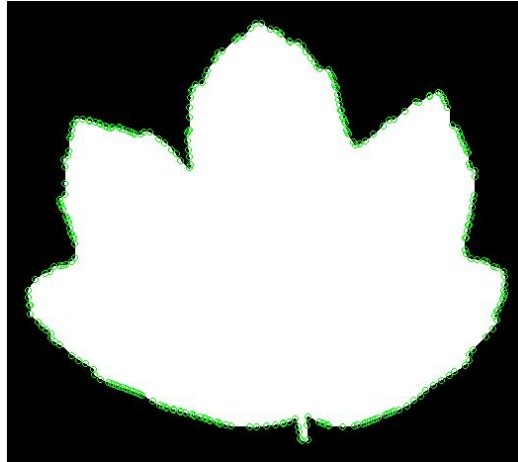


Figura 3 - “Exemplo pontos relevantes”

Os **inputs** dados à rede os **N pontos** (sendo N um valor dinâmico que varia com a escala de redimensionamento) **mais importantes relevantes** do contorno da folha.

Mas antes de obtermos esses pontos, todas as imagens têm que ser normalizadas de forma a existir uma norma.

(As imagens das folhas foram-nos fornecidas em formato “.jpg” e em preto em branco.)

Sempre que uma imagem é carregada esta passa por **3 passos**:

4.1.1 Rotação

Garantir que as imagens têm **a mesma orientação**, nós optámos por ter uma orientação vertical.

Apenas as imagens que se encontram deitadas são rodadas. Este processo examina o comprimento e a largura da folha, tendo um parâmetro de tolerância.

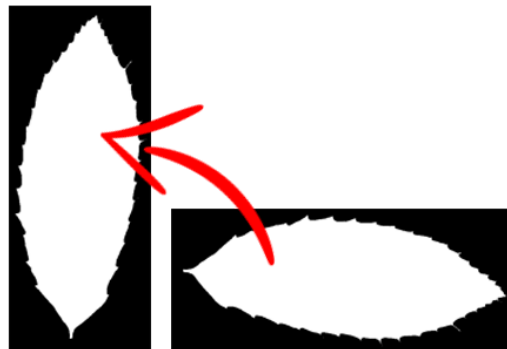


Figura 3 - “Exemplo Rotação das imagens”

4.1.2 Redimensionamento

Após rodadas, todas as imagens são redimensionadas e centradas numa “moldura” de tamanho $N \times N$ (sendo N uma variável dinâmica que determina o tamanho da “moldura”).

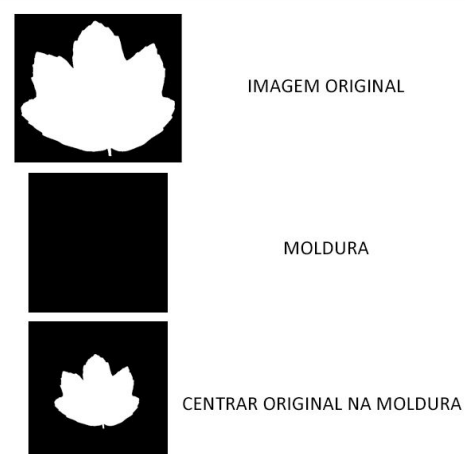


Figura 3 - “Exemplo Redimensionamento da imagem”

4.1.3 Conversão para Dados

4.1.3.1 Extração de pontos

Finalmente após Normalizada é feita a extração de N pontos mais relevantes da folhas a partir da “Computer Vision System Toolbox”.

```
pontos = detectHarrisFeatures(IMAGEM);  
pontos = points.selectStrongest(N).Location;
```

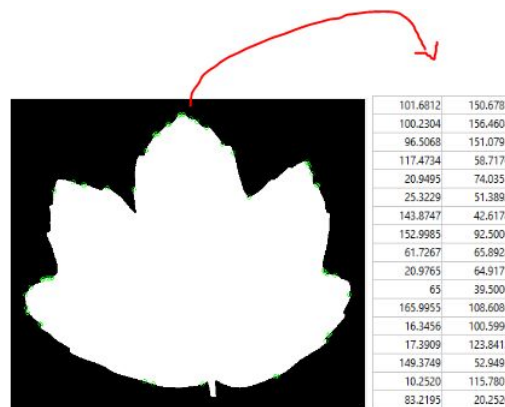


Figura 4 “Pontos Mais relevantes”

4.1.3.2 Matriz Binária

(Este método foi usado na abordagem inicial assim como para comparação de dados e resposta a algumas perguntas do Enunciado, não é no entanto a conversão usada para a rede neuronal final)

Finalmente após a rotação e a normalização das imagens, converteu-se as imagens para uma matriz binária, em que cada pixel da imagem é representado por 0 (Pretos) e 1 (Brancos).




Figura 5 - “Passagem de imagem para array binário”

4.2 Tratamento dos Targets

O **target** da rede neuronal final é um **código de identificação da Subespécie**, existe no entanto um código de representação de espécie.

Nota: os códigos das folhas estão organizados por ordem alfabética, como tal a representação

1	0	0	...	0	Código representativo da primeira Espécie/SubEspécie
...	
0	0	0	...	1	Código representativo da última Espécie/SubEspécie



	1	2	3
1	1	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0

	1	2	3
49	0	0	0
50	0	0	0
51	0	0	0
52	0	0	0
53	0	0	0
54	0	0	0
55	1	0	0
56	0	0	0
57	0	0	0

Figura 6 - "Código das folhas"

5- Configurações dos treinos

5.1 Funções de treino

- **trainrp** - Resilient backpropagation;
- **traingd** - Gradient descent backpropagation;
- **traingfb** - BFGS(é um algoritmo) quasi-Newton backpropagation;
- **trainlm** - Levenberg-Marquardt backpropagation.
- **trainoss** - One-step secant backpropagation
- **trainscg** - Scaled conjugate gradient backpropagation

5.2 Funções de ativação

- **tansig** - função de transferência hiperbólica, tangente , sigmóide;
- **logsig** - função de transferência sigmóide;
- **radbasn** - função de transferência de base radial normalizada.

5.3 Topologias

- **feedforwardnet** - As feedforward networks podem ser usadas para qualquer tipo de input para output mapeamento.
- **cascadeforwardnet** - Cascade-forward networks são parecidas com as feedforward networks, mas inclui uma conexão do input das camadas anteriores às camadas seguintes.
- **patternnet** - Pattern recognition networks são feedforward networks que podem ser treinadas para classificar inputs de acordo com o target class.
- **fitnet** - fitnet é um processo de treino da rede neuronal com um conjunto de inputs com a finalidade de produzir um conjunto associado de “targets”.

6- Exercício A

Para este exercício foi pedido a implementação de uma rede neuronal capaz de classificar as imagens espécie numa fase inicial e posteriormente conseguir determinar a sua subespécie usando como input uma matriz binária (representação da folha).

Apesar de termos usado como Input os pontos mais relevantes da folha na versão final do programa, para a **alínea A** e os seguintes resultados **usamos** a **Matriz Binária** como o **Input** pedido.

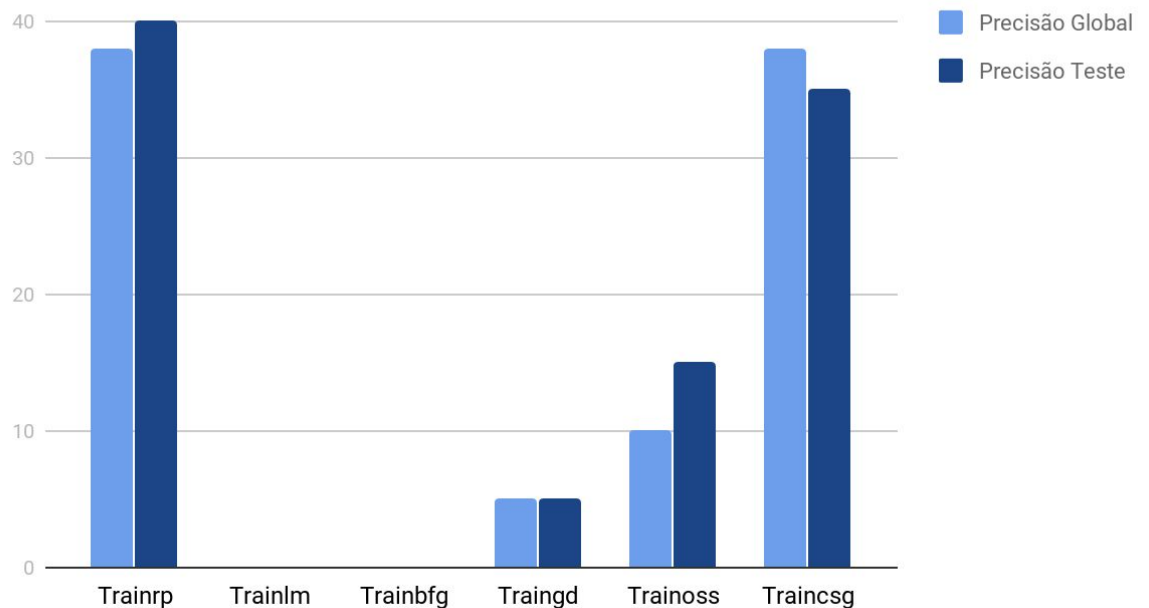
6.1- Implementação de uma rede neuronal para classificar por espécies

6.1.1 Feedforwardnet

Funções de treino

- Para 6 configurações com:
 - Número de camadas escondidas - 2;
 - Número de neurónios -10;
 - Funções de ativação - 'tansig' (para ambas as camadas);
 - Divisão de exemplos - dividerand {0.4;0.4;0.2};
 - Topologia - feedforwardnet.
- Variando apenas o atributo função de treino obteve-se o seguinte gráfico

Função de treino

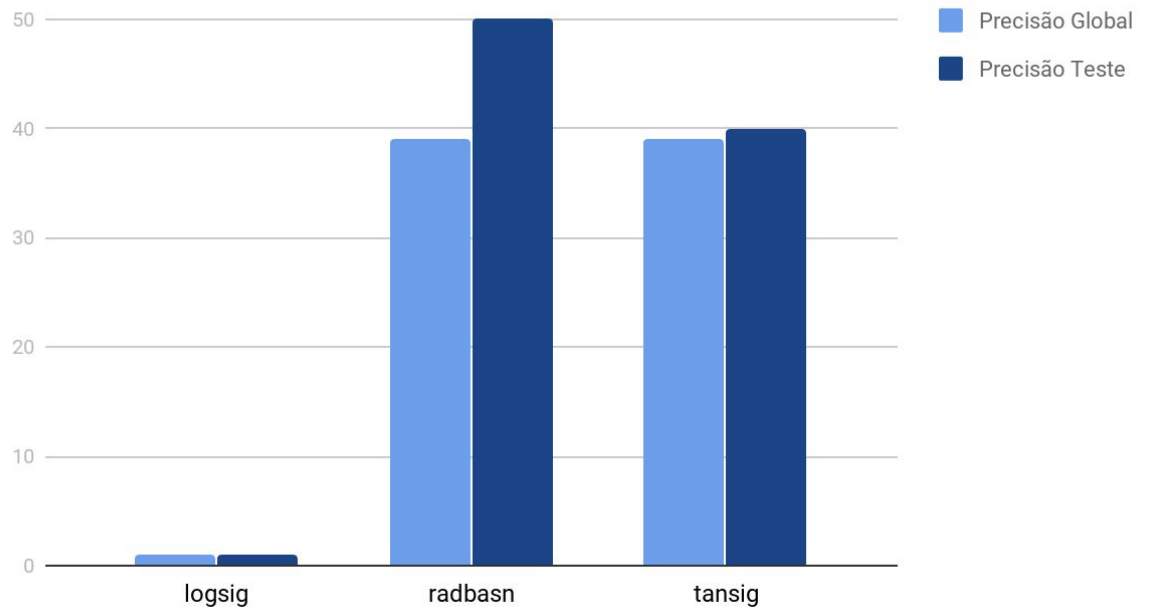


Como podemos concluir com o gráfico anterior com a topologia feedforwardnet, os melhores resultados obtidos foram com a função de treino “trainrp”, tendo as funções “trainlm” e “trainbfg” terem dado erro de “array size limit” devido à computação ser mais elevada, e a função “traingd” prova ser a mais fraca.

Função de ativação

- Para 3 configurações com:
 - Número de camadas escondidas - 2;
 - Número de neurónios -10;
 - Funções de treino - “trainrp”;
 - Divisão de exemplos - dividerand {0.4;0.4;0.2};
 - Topologia - feedforwardnet.
- Variando apenas o atributo função de ativação (a mesma para ambas as camadas) obteve-se o seguinte gráfico:

Função de ativação



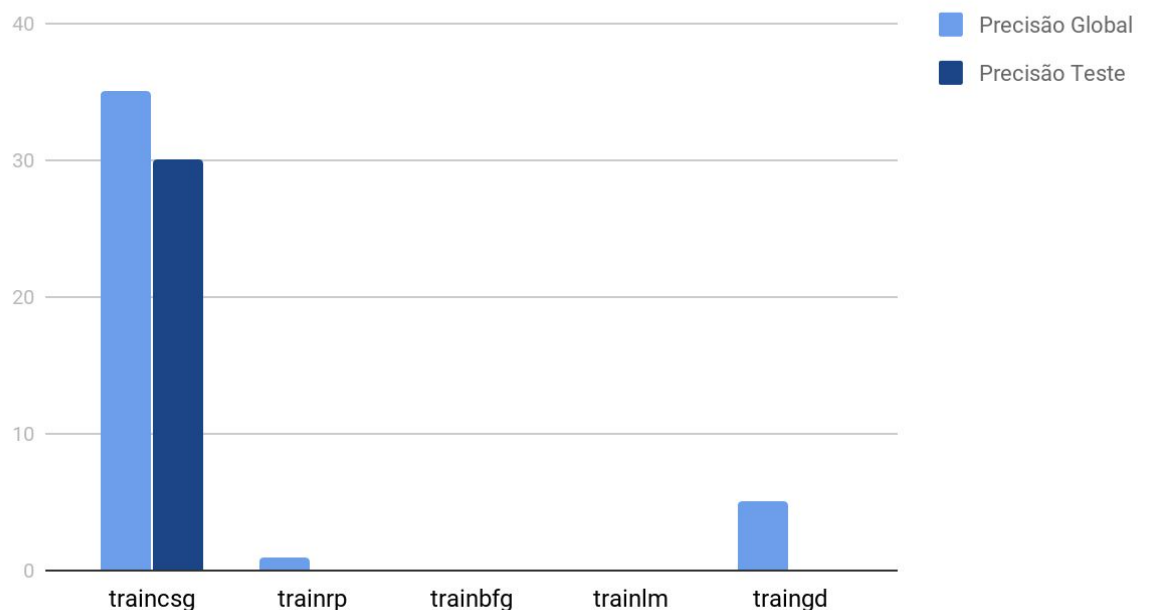
Por observação concluímos que as funções de ativação que treinam melhor a rede neuronal são “radbasn” e “tansig”, e a função “logsig” obteve valores de precisão mais baixos.

6.1.2 Patternnet

Funções de treino

- Para 5 configurações com:
 - Número de camadas escondidas - 2;
 - Número de neurónios -10;
 - Funções de ativação - ‘tansig’ (para ambas as camadas);
 - Divisão de exemplos - dividerand {0.4;0.4;0.2};
 - Topologia - patternnet.
- Variando apenas o atributo função de treino obteve-se o seguinte gráfico:

Patternnet (função de treino)



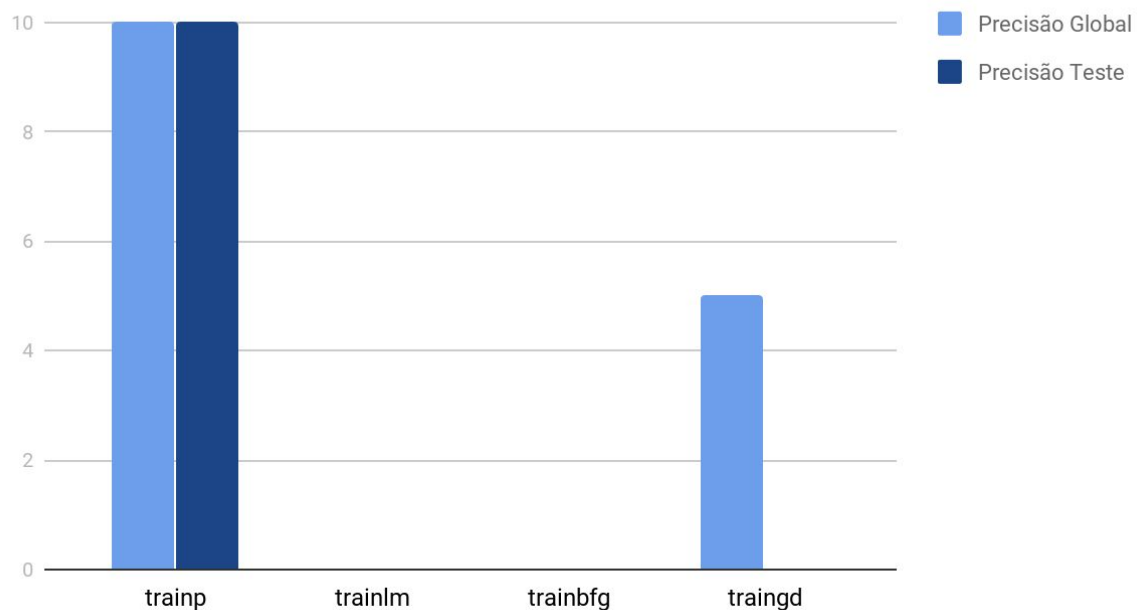
Por observação do gráfico podemos concluir que a função de treino, “*traincsg*” da topologia “*patternnet*” é a que consegue obter, por grande margem, melhores resultados, excluindo assim as funções “*trainlm*” e “*trainbfg*” que dão erro de “*array size limit*”.

6.1.3 Cascadeforwardnet

Funções de treino

- Para 4 configurações com:
 - Número de camadas escondidas - 2;
 - Número de neurónios - 10;
 - Funções de ativação - 'tansig' (para ambas as camadas);
 - Divisão de exemplos - dividerand {0.4;0.4;0.2};
 - Topologia - cascadeforwardnet.
- Variando apenas o atributo função de treino obteve-se o seguinte gráfico:

Cascadeforwardnet (função de treino)



Por observação ao gráfico pode-se concluir que todas as configurações deram resultados de precisão muito baixos, excluindo as funções “trainlm” e “trainbfg” que dão erro de “array size limit”.

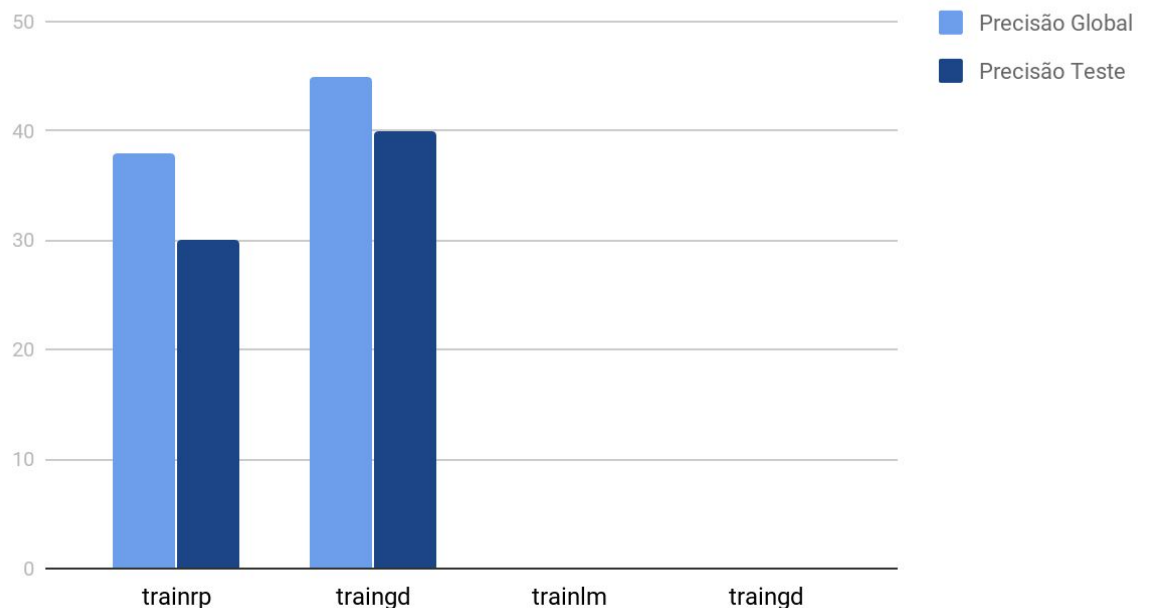
6.1.4 Fitnet

Funções de treino

- Para 4 configurações com:
 - Número de camadas escondidas - 2;
 - Número de neurónios -10;
 - Funções de ativação - ‘tansig’ (para ambas as camadas);
 - Divisão de exemplos - dividerand {0.4;0.4;0.2};
 - Topologia - fitnet.

- Variando apenas o atributo função de treino obteve-se o seguinte gráfico

Fitnet (função de treino)



Após observação do gráfico, podemos concluir que as funções de treino “trainlm” e “traingd” continuam a dar erro de “array size limit” e que as funções “train, traingd” ambas apresentam uma boa taxa de precisão.

Conclusão

Ao estudarmos os dados e os resultados apercebemo- nos que o tempo e os recursos de processamento entre topologias e funções varia imenso, a questão que ficou foi *“Será que é proporcional ao tamanho do DataSet ou será da algoritmia de cada topologia/Função, será que o tipo de Input influencia drasticamente?”*. Para respondermos a tais perguntas são necessários mais testes, pois estes dados são demasiados pequenos para se conseguir tirar conclusões genéricas.

No entanto, certas coisas sobressaíram como por exemplo, de todas as **topologias** apenas a **Cascadeforwardnet** apresenta soluções extremamente **baixas** e que das funções de treino a **função trainrp** se destaca também mas por valores **altos**, e se tivermos como **Input** matriz binária da imagem e como **target** apenas a espécie a **melhor solução** será **Feedforwardnet** com a função de treino **trainrp** e a função de ativação **radbasn**.

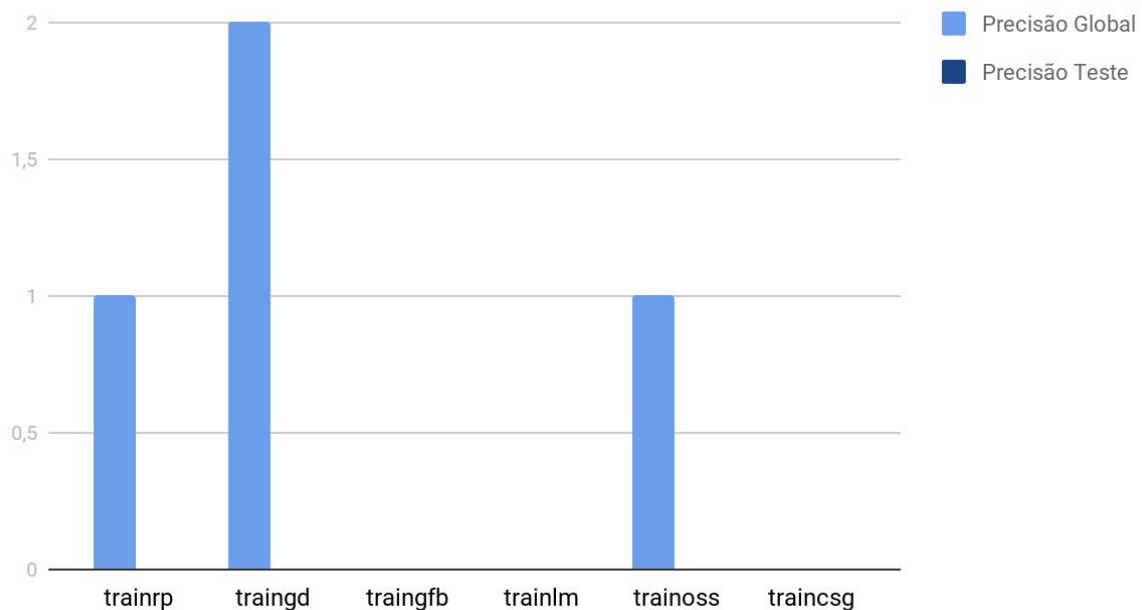
6.2 Implementação de uma rede neuronal para classificar por subespécies.

6.2.1 Feedforwardnet

Função de treino

- Para 6 configurações com:
 - Número de camadas escondidas - 2;
 - Número de neurónios -10;
 - Funções de ativação - 'tansig' (para ambas as camadas);
 - Divisão de exemplos - dividerand {0.4;0.4;0.2};
 - Topologia - feedforwardnet.
- Variando apenas o atributo função de treino obteve-se o seguinte gráfico

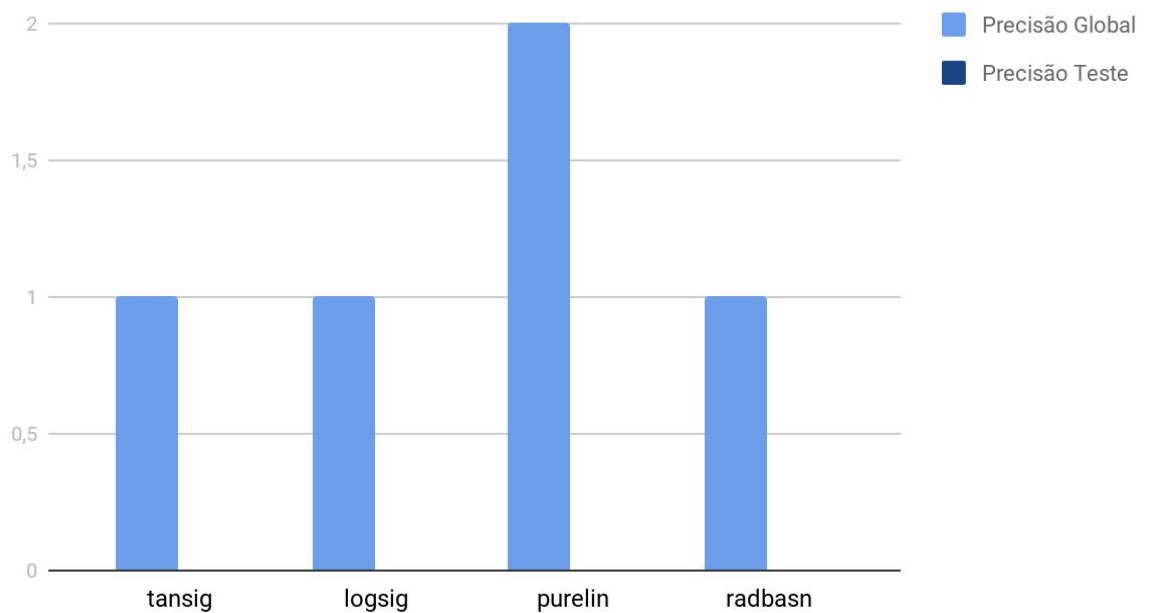
Função de treino



Função de ativação

- Para 4 configurações com:
 - Número de camadas escondidas - 2;
 - Número de neurónios -10;
 - Funções de treino- trainrp;
 - Divisão de exemplos - dividerand {0.4;0.4;0.2};
 - Topologia - feedforwardnet.
- Variando apenas o atributo função de ativação em ambas as camadas obteve-se o seguinte gráfico

Função de ativação

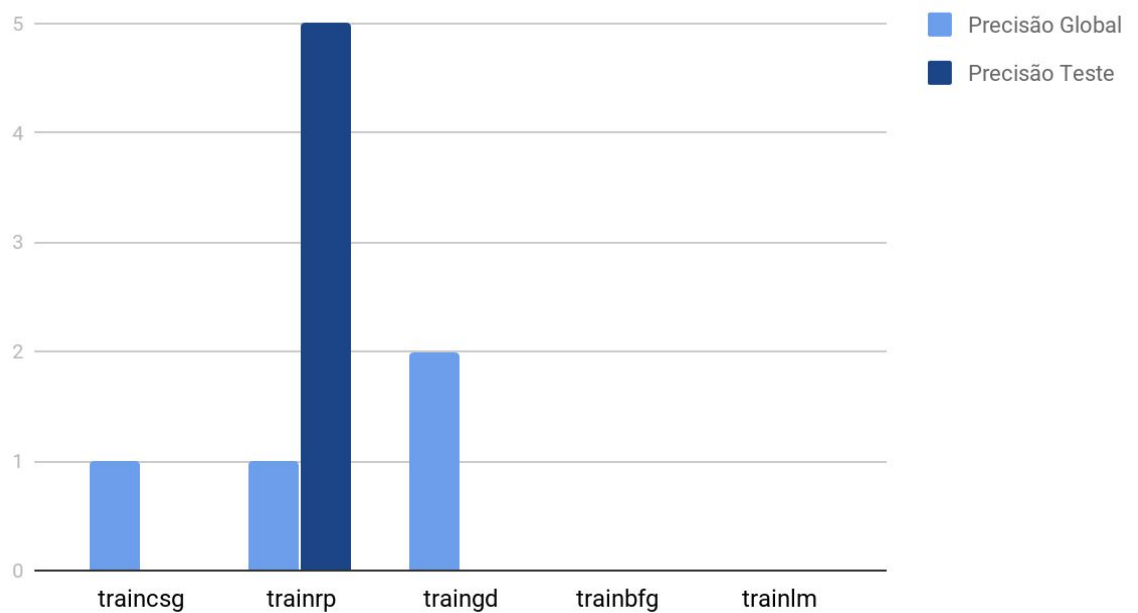


6.2.2 Patternnet

Função de treino

- Para 6 configurações com:
 - Número de camadas escondidas - 2;
 - Número de neurónios -10;
 - Funções de ativação - 'tansig' (para ambas as camadas);
 - Divisão de exemplos - dividerand {0.4;0.4;0.2};
 - Topologia -patternnet.
- Variando apenas o atributo função de treino obteve-se o seguinte gráfico

Patternnet (função de treino)

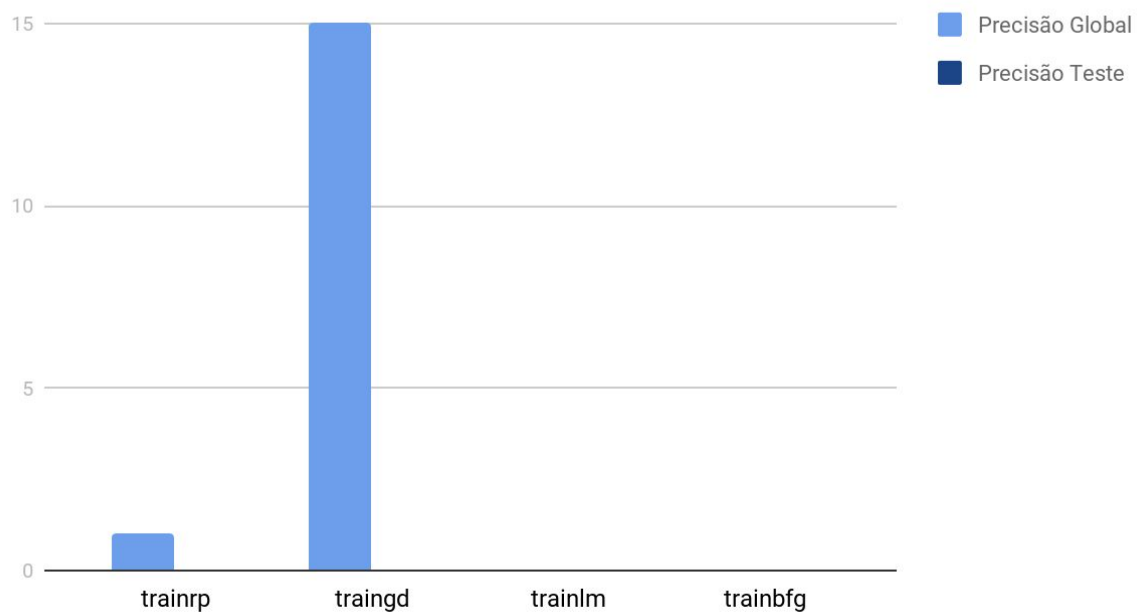


6.2.3 Cascadeforwardnet

Função de treino

- Para 6 configurações com:
 - Número de camadas escondidas - 2;
 - Número de neurónios -10;
 - Funções de ativação - 'tansig' (para ambas as camadas);
 - Divisão de exemplos - dividerand {0.4;0.4;0.2};
 - Topologia - cascadeforwardnet.
- Variando apenas o atributo função de treino obteve-se o seguinte gráfico

Cascadeforwardnet (função de treino)

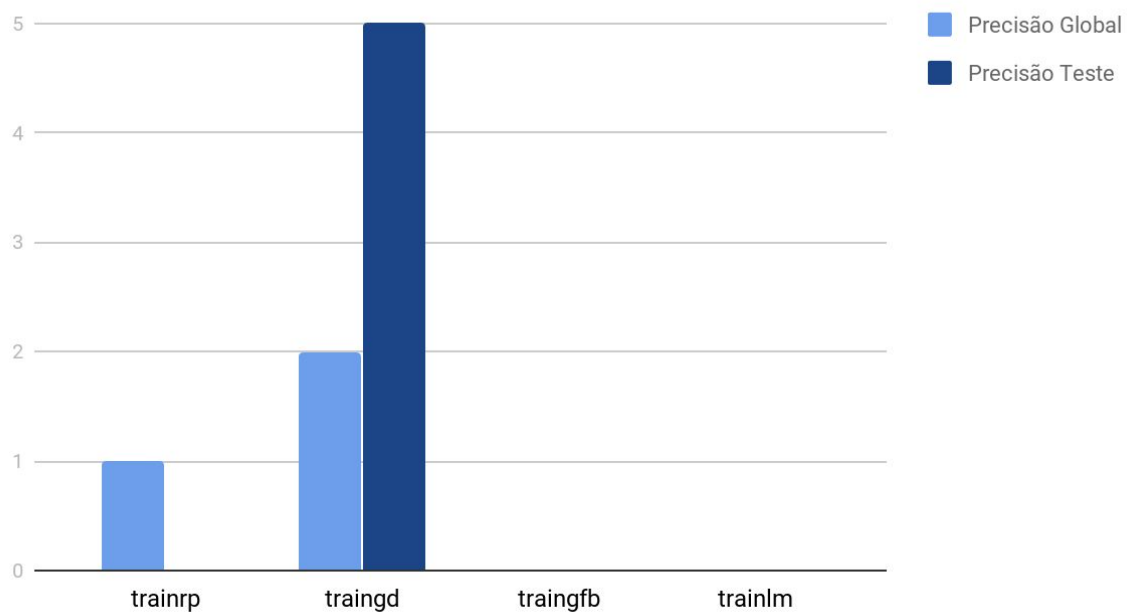


6.2.4 Fitnet

Função de treino

- Para 6 configurações com:
 - Número de camadas escondidas - 2;
 - Número de neurónios -10;
 - Funções de ativação - 'tansig' (para ambas as camadas);
 - Divisão de exemplos - dividerand {0.4;0.4;0.2};
 - Topologia - fitnet.
- Variando apenas o atributo função de treino obteve-se o seguinte gráfico

Fitnet (função de treino)



Conclusão

Ao estudarmos os dados e os resultados apercebemo-nos que, **apesar** da **lógica** para determinar a Espécie e a SubEspécie **ser a mesma, os resultados não** são semelhantes, concluindo assim que, para conseguirmos resultados melhores é **necessário** alterar algo. Existem várias coisas que podem ser **modificadas**:

- **Input**
 - Modificar a representação da Imagem
- **Output**
 - Segmentar as Subespécies por Espécies
- **Rede Neuronal**
 - Ter diversas redes cada uma responsável por uma característica diferente

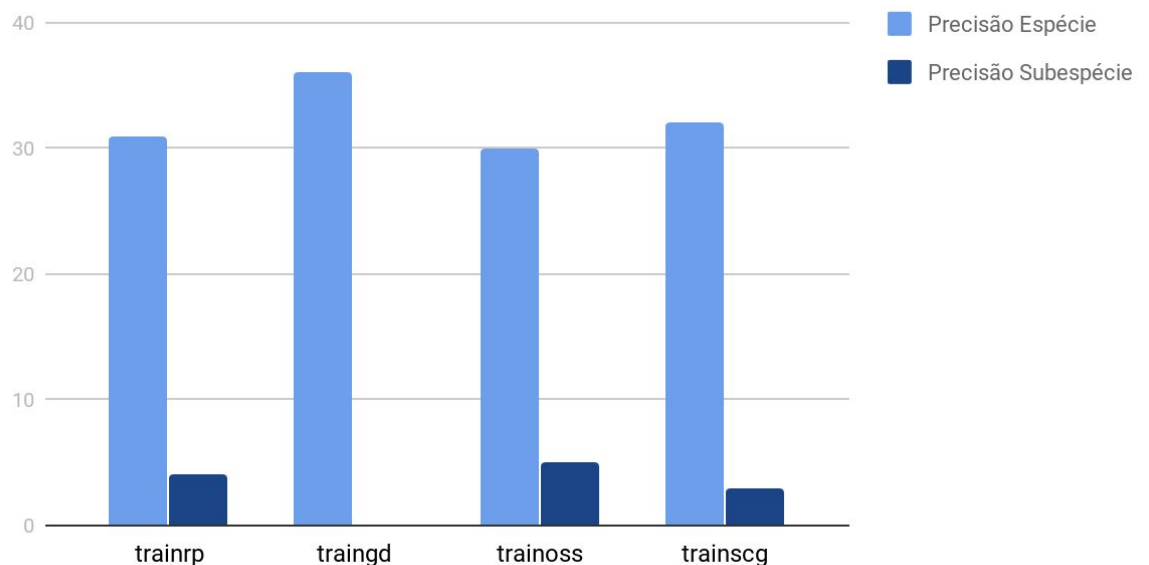
7- Exercício B

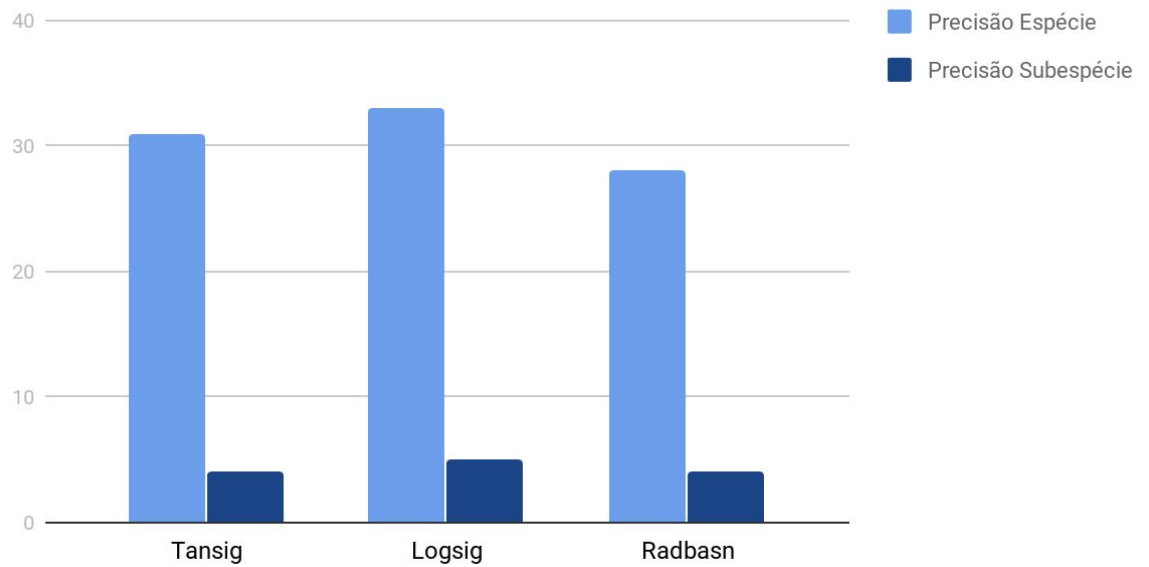
Para este exercício foi nos pedido que repetíssemos o método da alínea A, garantido apenas que os parâmetros de segmentação do dataset fossem: [70%, 15%, 15%], no então, após concluirmos que a matriz era um **Input demasiado fraco** por si só, nós **mudámos** o **input** parâmetros e não a representação binária pura.

7.1 Feedforwardnet

Funções de treino

Ex B FREEDFORWARD (FUNÇÃO DE TREINO)
Abordagem por parâmetros



Funções de ativação**Ex B FEEDFORWARD (FUNÇÃO DE ATIVAÇÃO)**
Abordagem por parâmetros

Matrix Confusão

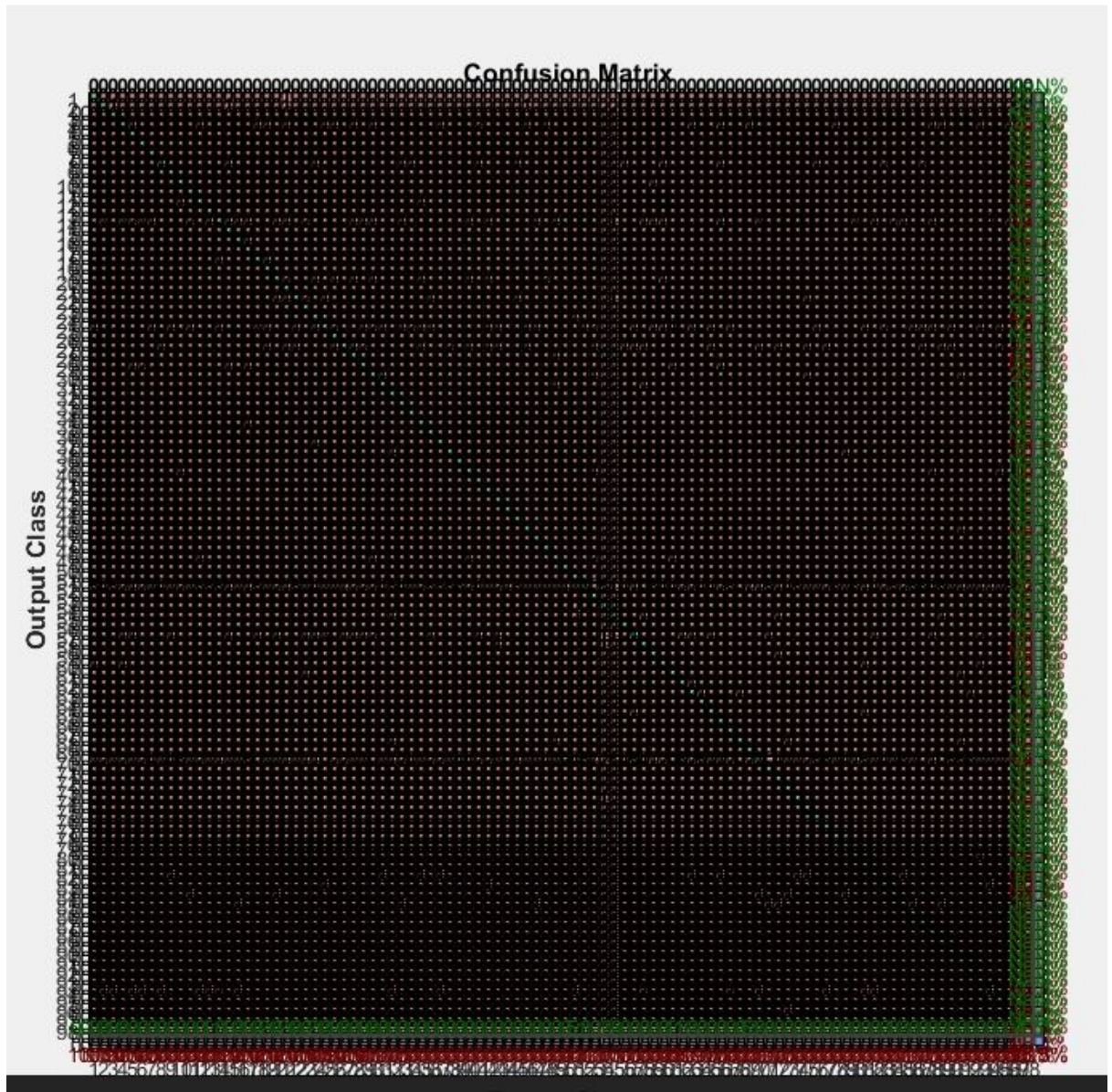


Figura 7 - “Matriz de confusão feedforwardnet”

Testes

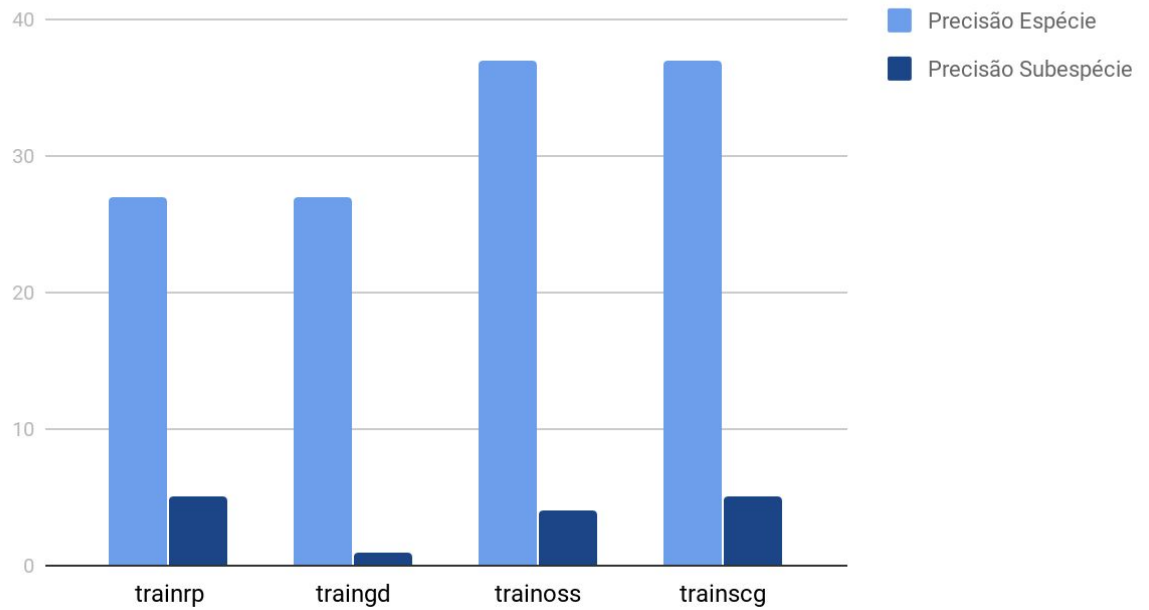
- Usando a melhor rede obtida:
 - **resultados:**

% Espécies Correctas	% SubEspecies Correctas
33%	5%

7.2 Fitnet

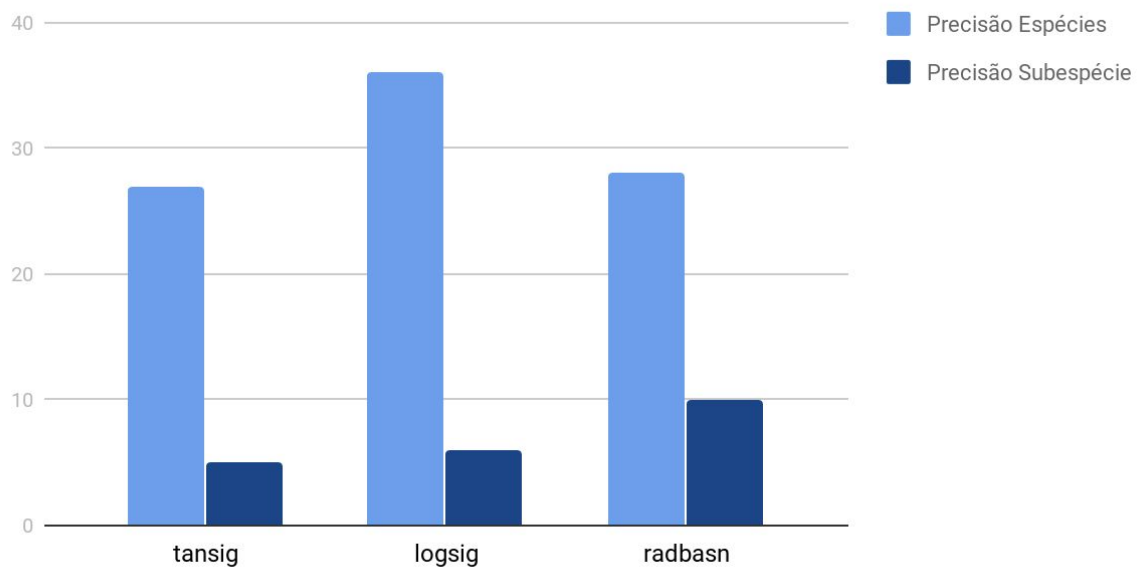
Funções de treino

Ex B FITNET (FUNÇÃO DE TREINO)



Funções de ativação

Ex B FITNET (FUNÇÃO DE ATIVAÇÃO)
abordagem por parametros



Matrix Confusão

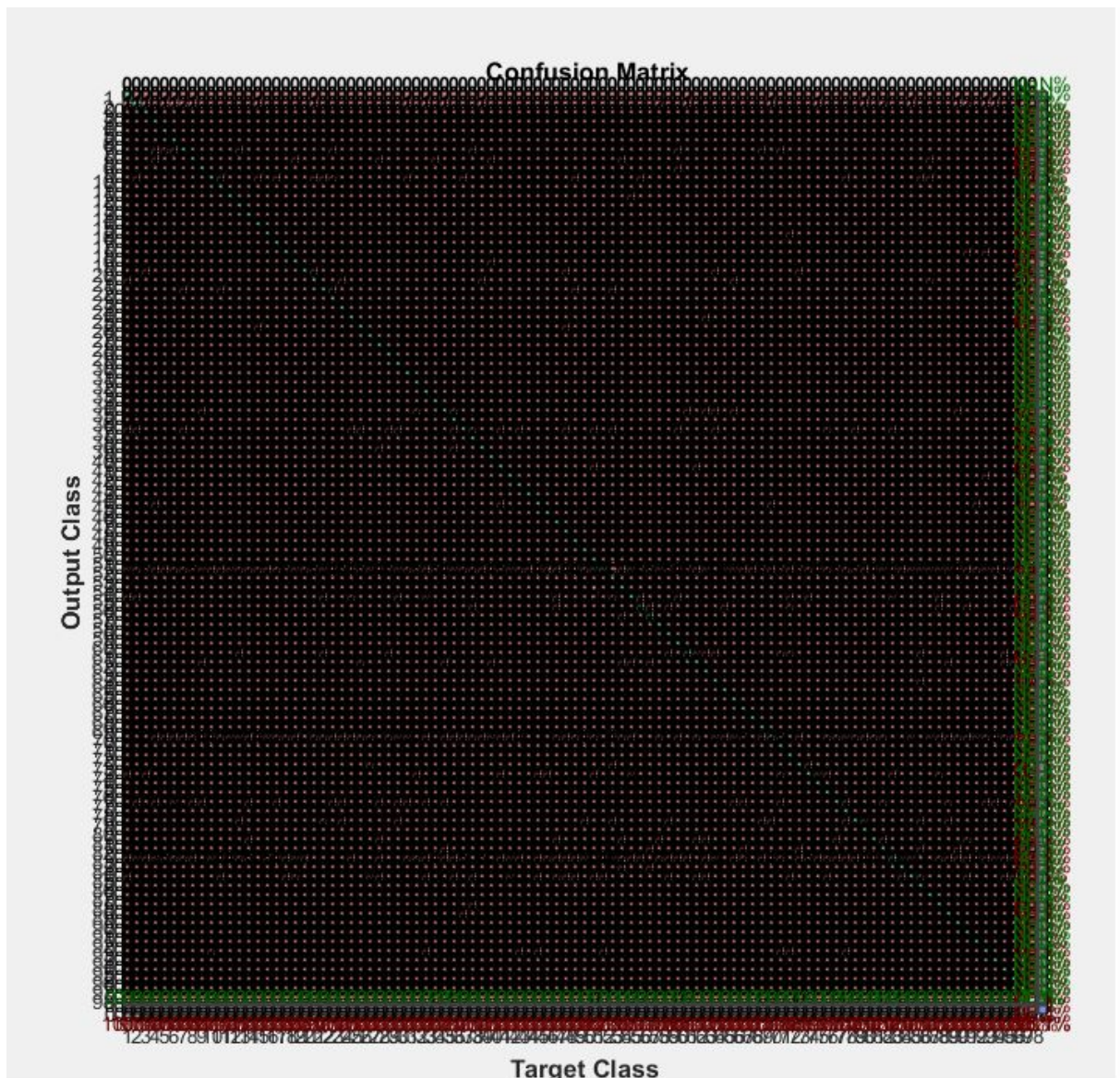


Figura 8 - "Matriz de confusão fitnet"

Testes

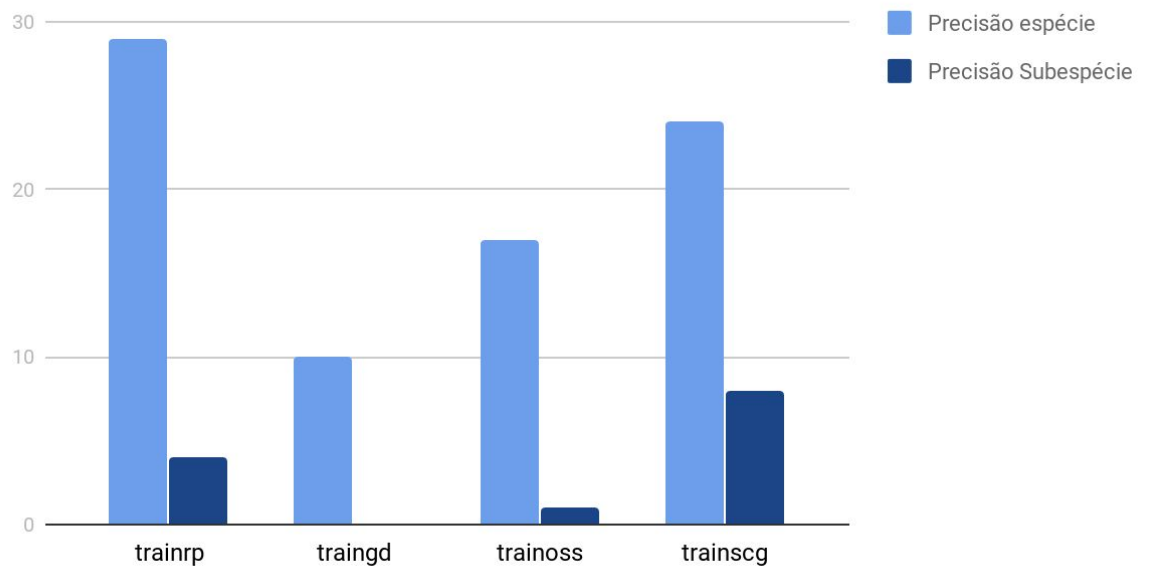
- Usando a melhor rede obtida:
 - **resultados:**

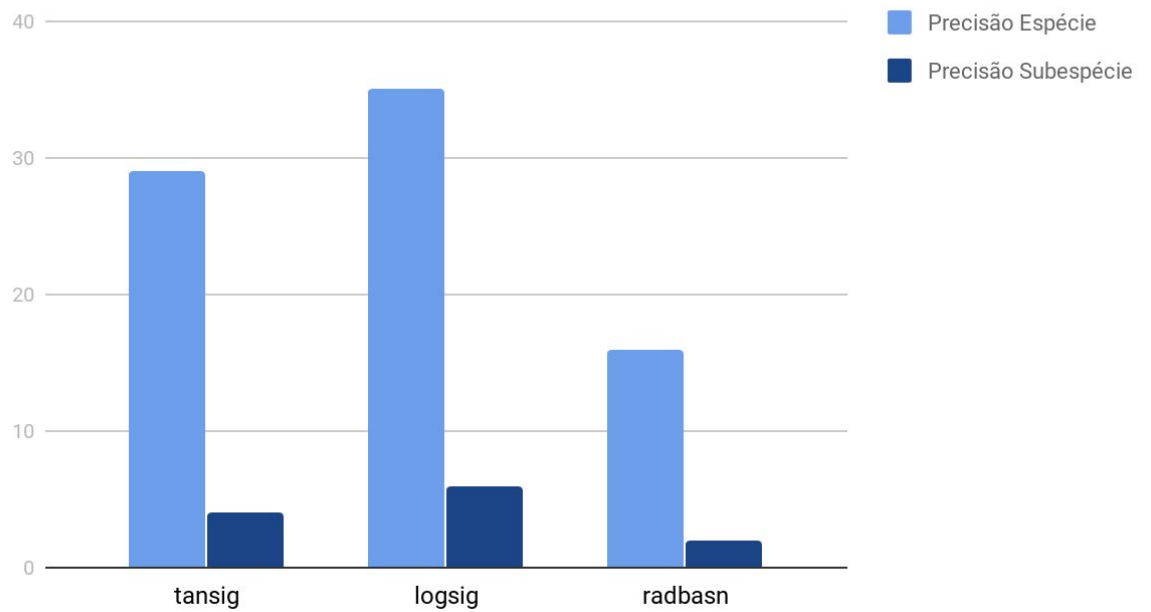
% Espécies Correctas	% SubEspecies Correctas
37%	5%

7.3 Patternnet

Funções de treino

Ex B PATERNNET (FUNÇÃO DE TREINO)
abordagem por parametros



Funções de ativação**Ex B PATERNNET (FUNÇÃO DE ATIVAÇÃO)**

Matrix Confusão

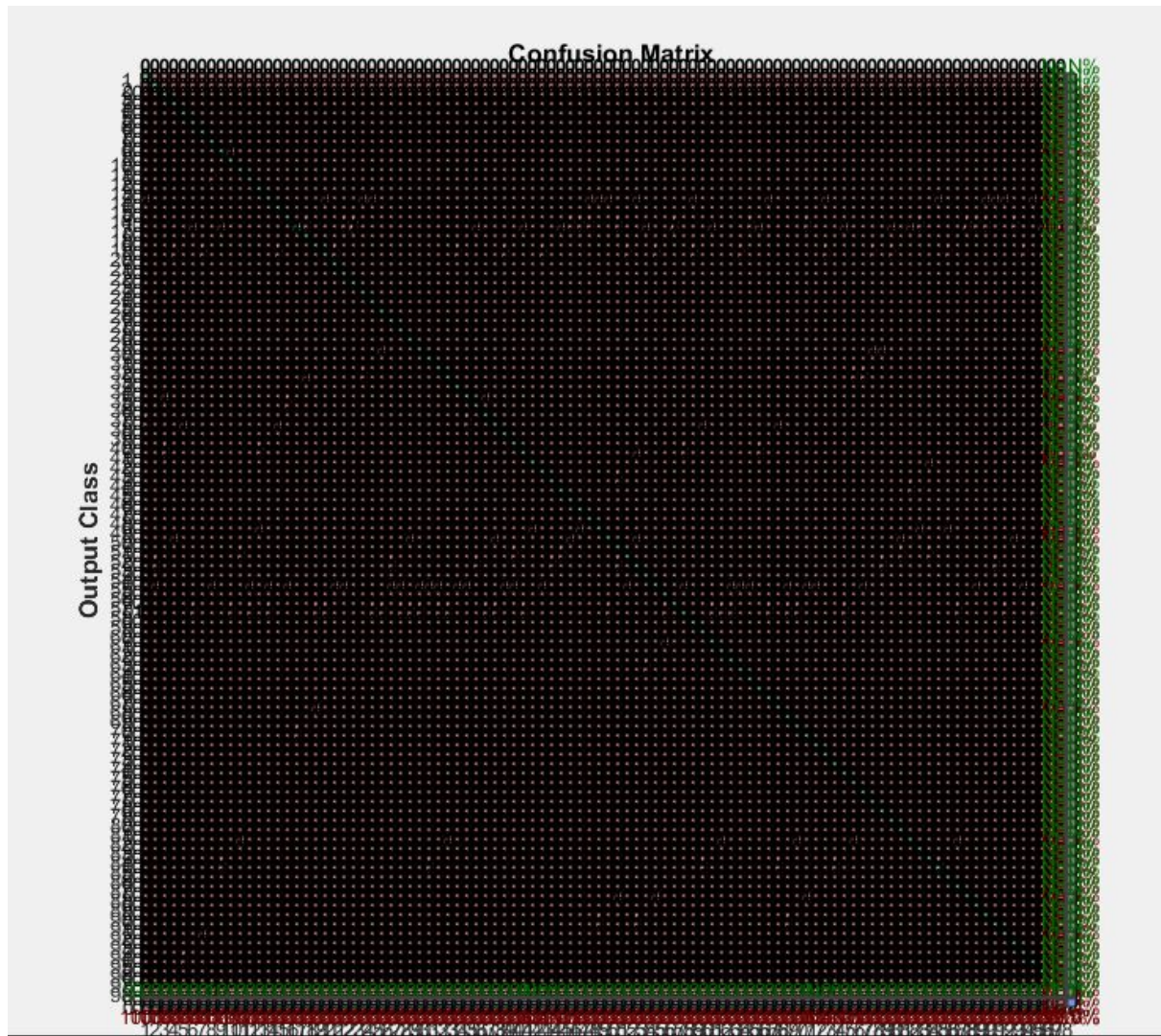


Figura 9 -“Matriz de confusão paternnet”

Testes

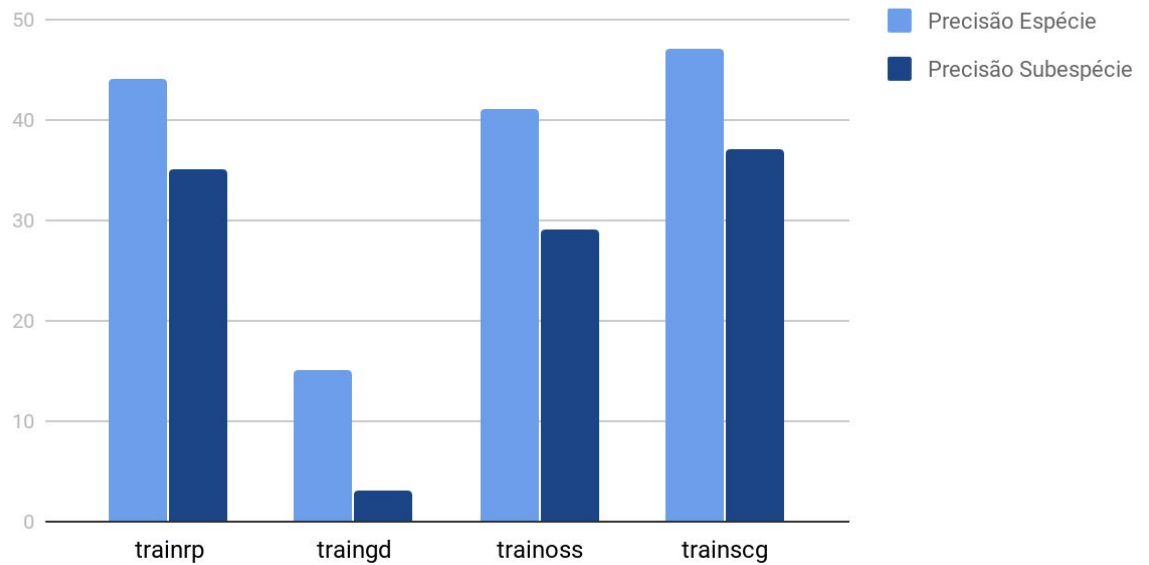
- Usando a melhor rede obtida:
 - **resultados:**

% Espécies Correctas	% SubEspecies Correctas
44%	35%

7.4 Cascadeforwardnet

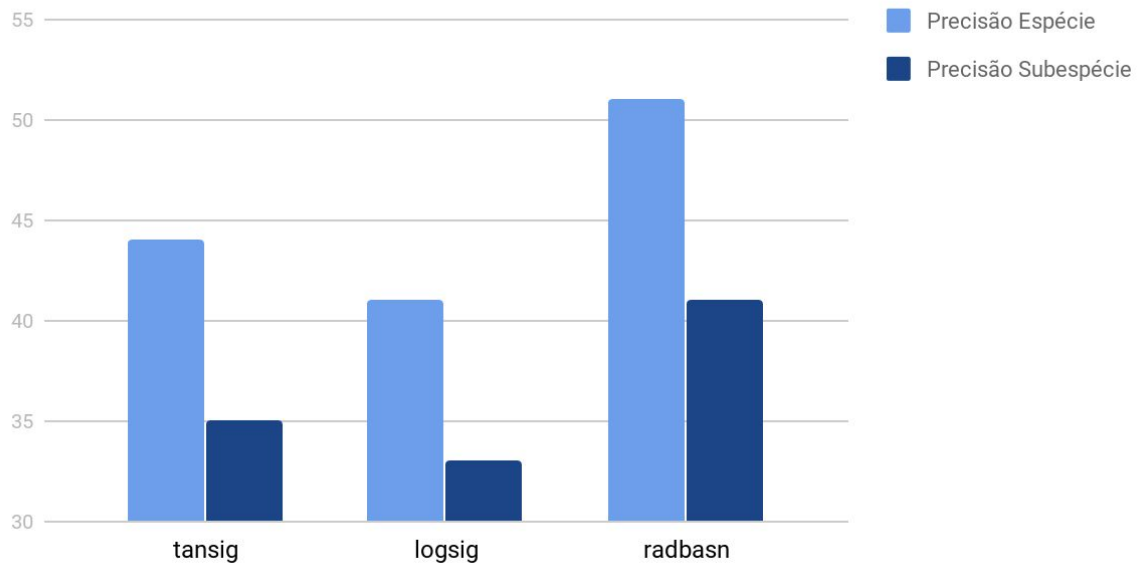
Funções de treino

Ex B CASCADEFORWARDNET (FUNÇÃO DE TREINO)
abordagem por parametros



Funções de ativação

Ex B CASCADEFORWARDNET (FUNÇÃO DE ATIVAÇÃO)
abordagem por parametros



Matrix Confusão

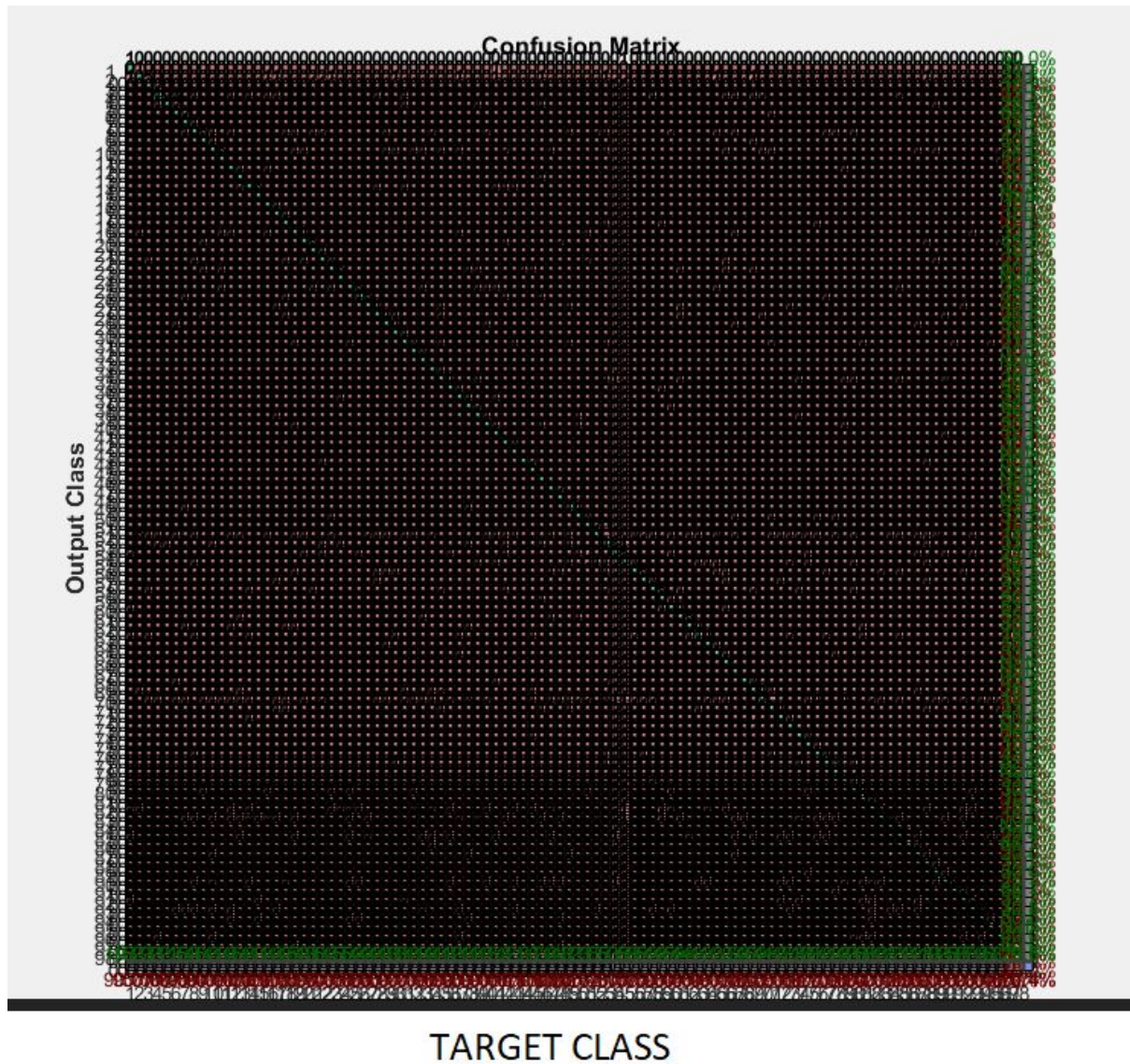


Figura 10 - "Matriz de confusão cascadedforwardnet"

Testes

- Usando a melhor rede obtida:
 - **resultados:**

% Espécies Correctas	% SubEspecies Correctas
51%	41%

Conclusão

Devido a haver um leque **maior** de **possibilidades**, existe também uma **dificuldade acrescida** em criar uma solução que baseado num dataset mais simples produza o mesmo nível de resultados, é aqui que notamos a diferença numa boa arquitetura de uma fraca, e acima de tudo a diferença que o dataSet de treino tem perante os resultados **em situações desconhecidas**, apesar de ter tido uma precisão mais baixa que a rede da alinha A, **esta rede está mais bem equipada para lidar com situações desconhecidas, uma vez que tem mais experiências únicas.**

8- Exercício C

- Usando a melhor rede do data set 1, com uma precisão de 71%:
 - **resultados:**

% Espécies Correctas	% SubEspecies Correctas
15%	0

- Usando a melhor rede do data set 2 com uma precisão de 49%:
 - **resultados:**

% Espécies Correctas	% SubEspecies Correctas
15%	5%

- Rede treina com o dataset 3 em mente.

% Espécies Correctas	% SubEspecies Correctas
95%	84%

(é de salientar que no entanto tem péssimos resultados em aplicações reais e nos outros dataset's)

Conclusão

Devido a **este dataset** ser altamente **específico**, em que algumas folhas têm pouca representação nas pastas nos datasets 1 e 2, podemos ver que as redes **treinadas anteriormente não são tão eficazes** aqui.

A rede que usou o dataset 2 mesmo tendo uma precisão relativa ao seu dataset de apenas 49% supera a rede que usou o dataset 1 de precisão 71%, isto pois o dataset 2 é mais abrangente e completo.

Isto comprova o que já tinha sido descoberto. que **os dataset's de forma a garantirem uma rede genérica, devem ser equilibrados e extensos.**

9- Exercício D

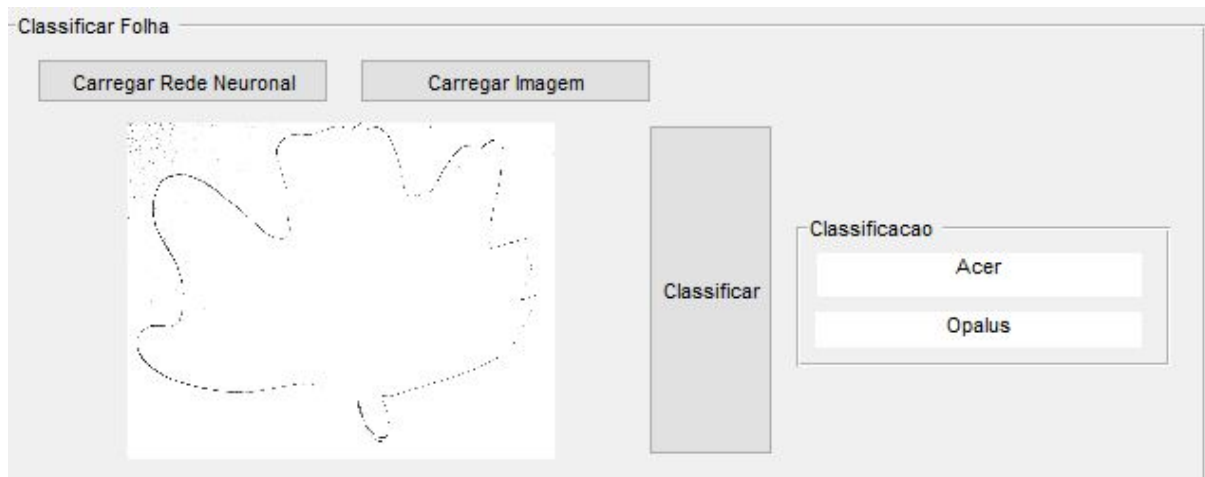


Figura 11 - “Classificar folha”

Apesar da **estrutura** do nosso programa estar **feita** para poder aceitar **qualquer tipo de imagem**, esta não é inteligente o suficiente para saber identificar o alvo como uma folha, independentemente do inserido, o algoritmo irá tentar encontrar a **resposta mais semelhante** e dará o resultado mais perto possível **por mais errada** que este **possa ser**. No **ponto 12 indicamos alternativas** possíveis assim como **melhoramentos** que possam vir a ser implementados de forma a garantir um resultado mais realista e correcto.

10- Exercício E

Como tem vindo a ser explicado ao longo deste relatório, o nosso algoritmo final foi modificado e melhorado para extração de features da imagem, de forma a conseguirmos resultados acima dos 40%, fizemos a **transição após** a análise de dados da **linha A** ser insuficiente. **Estamos a usar a extração de pontos** com maior relevância numa imagem.



Figura 12 - “extração de pontos”

No âmbito do trabalho prático e do dataset fornecido, chegamos à conclusão que **a extração de pontos é suficiente para resultados com precisão elevada a um custo reduzido.**

No entanto, para imagens a cores e em que as folhas não estão a ser isoladas, a extração de mais características é fundamental para otimizar resultados.

11- Exercício F

Treinar redes neurais

Como pode ser facilmente visualizado pela imagem mais abaixo, as redes podem ser facilmente configuradas, algumas das funcionalidades são:

- **Configurar a topologia das redes neurais**
- **Configurar Hidden Layers/Neurônios**
- **Escolher funções de treino / ativação**
- **Gravar uma rede neuronal treinada**

The image shows a software interface titled "Treinar Redes". It contains several configuration options:

- Escolher Topologia:** A dropdown menu showing "feedforwardnet".
- Escolher DataSet:** A dropdown menu showing "Folhas_1".
- Escolher Funcoes Ativação:** A dropdown menu showing "tansig".
- HiddenLayers:** A numeric input field showing "1".
- Escolher Funcoes Treino:** A dropdown menu showing "trainrp".
- Neuronios:** A numeric input field showing "10".
- Precisao Teste:** An empty text input field.
- Precisao Total:** An empty text input field.
- Treinar Rede:** A button.
- Gravar Rede:** A button.

Figura 13 - "Painel de configuração do treino das redes"

Carregar uma rede neuronal

Como pode ser facilmente visualizado na imagem abaixo, as redes podem ser simuladas com os Dataset's fornecidos, algumas das funcionalidades do programa são:

- **Visualizar os resultados da classificação**
- **Geração/gravação dos resultados assim como as configurações da rede usada**

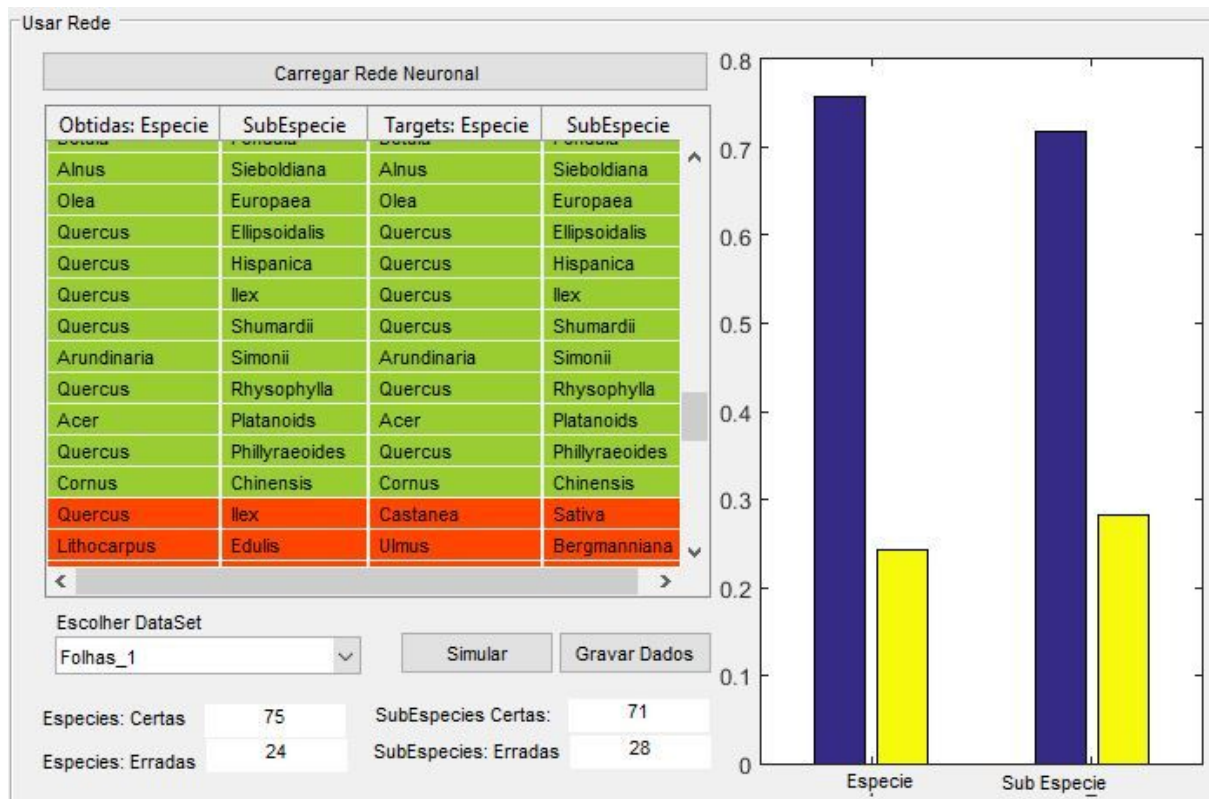


Figura 14 - "Simulação"

Aplicar uma rede neuronal para classificar folhas

O programa é capaz de classificar imagens individualmente, usando redes neurais previamente treinadas, assim como visualizar os **resultados da classificação**.

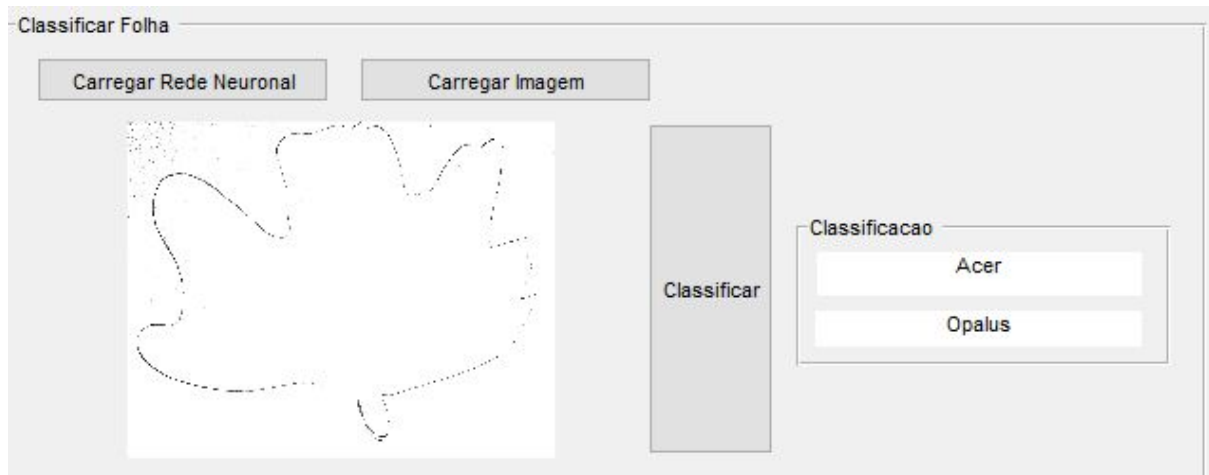


Figura 15 - "Classificar Imagem"

12- Irregularidades encontradas

DATASET

problema

Os dataset's fornecidos estavam altamente desnivelados, exemplo o dataset 1 inclui **33%** dos alvos como **espécie QUERCUS**. Este peso elevado de uma única espécie garante que uma rede facilmente obtenha precisão superior a 30% em espécies sem ser realmente útil noutros casos.

solução

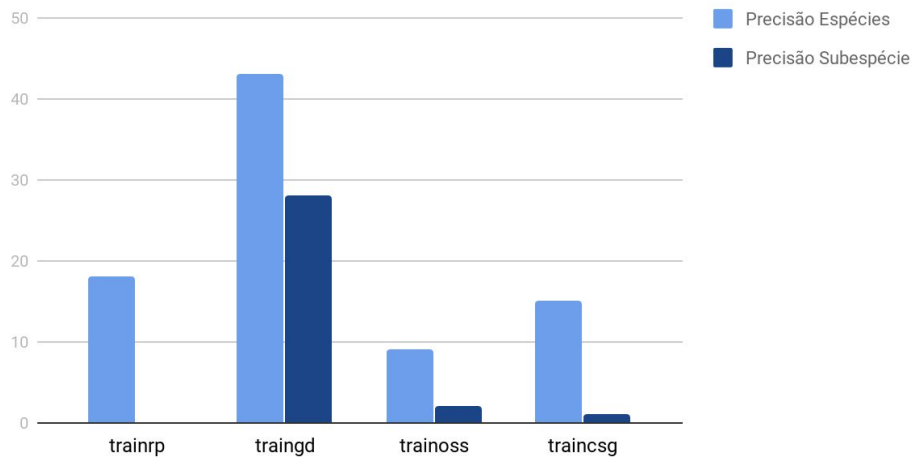
Para conseguirmos realmente criar uma rede neuronal que conseguisse obter resultados eficazes o **INPUT** pedido pela professora, a imagem em binário, tinha que ser fortemente manipulado. Abaixo deixamos a confirmação que não usando o INPUT pedido na alinha A facilmente conseguimos valores 6x superiores.

EXERCÍCIO A (feito com extração de dados)

12.1 Feedforwardnet

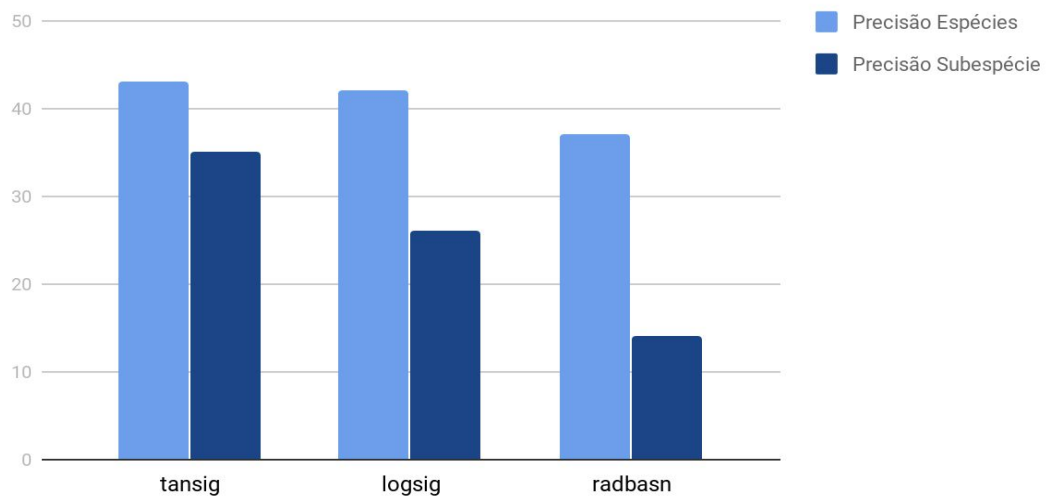
Funções de treino

EX 1 FEEDFORWARDNET (FUNÇÃO DE TREINO) abordagem por parametros



Funções de ativação

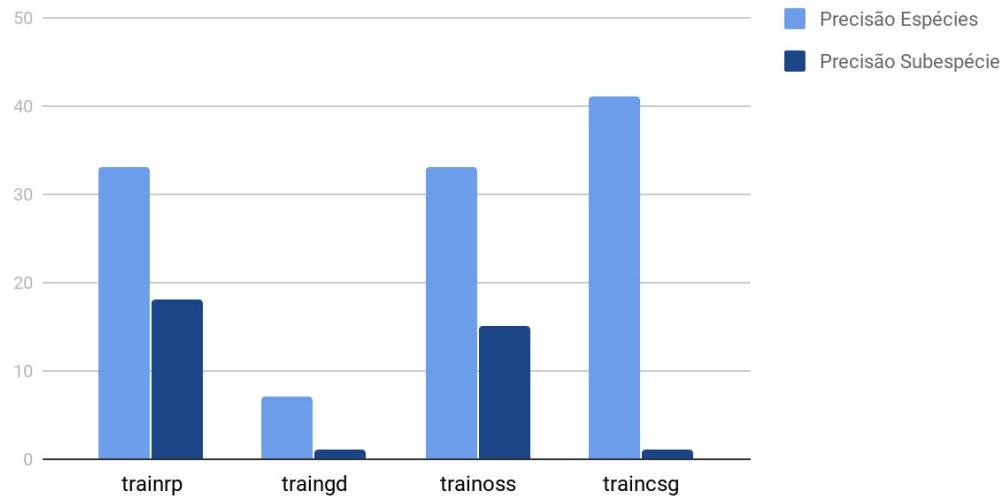
EX 1 FEEDFORWARDNET (FUNÇÃO DE ATIVAÇÃO) abordagem por parametros



12.2 Fitnet

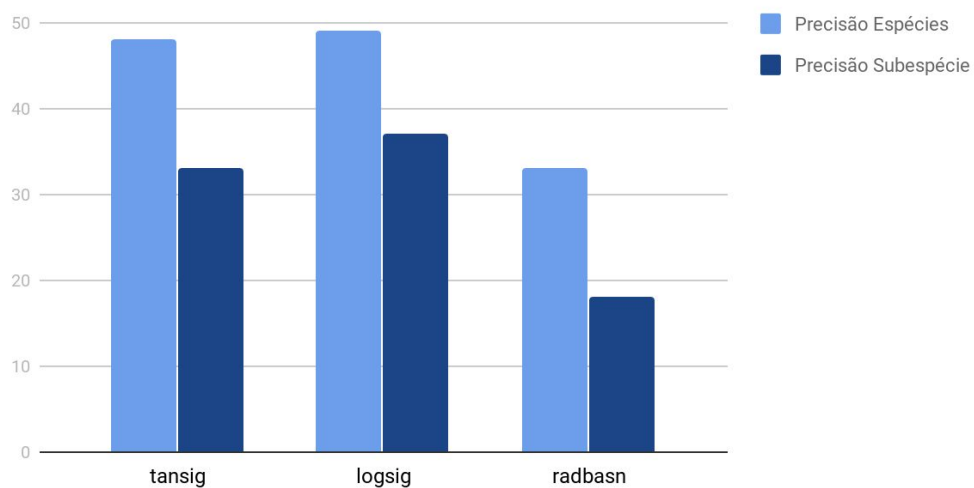
Funções de treino

EX1 FITNET (FUNÇÃO DE TREINO) abordagem por parametros



Funções de ativação

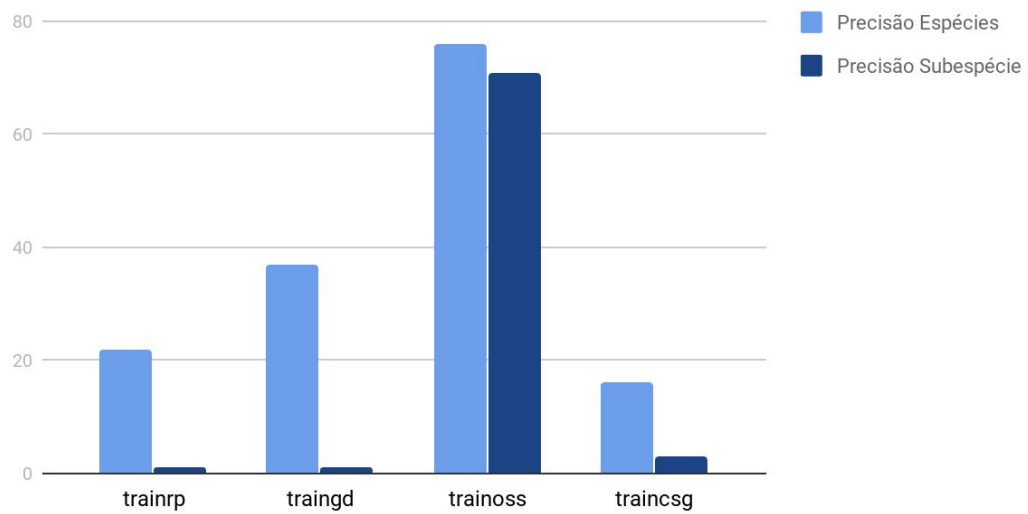
EX 1 FITNET (FUNÇÃO DE ATIVAÇÃO) abordagem por parametros



12.3 Patternnet

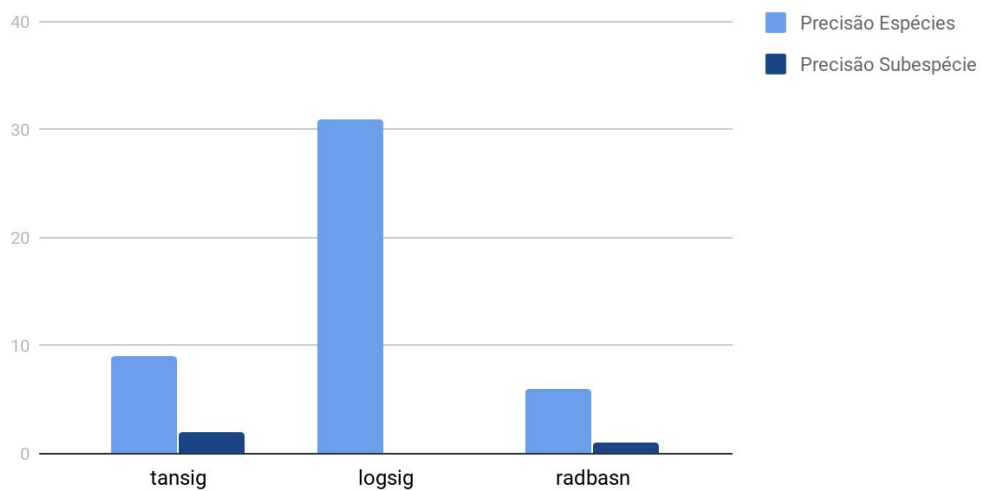
Funções de treino

EX 1 PATERNNET (FUNÇÃO DE TREINO) abordagem por parametros



Funções de ativação

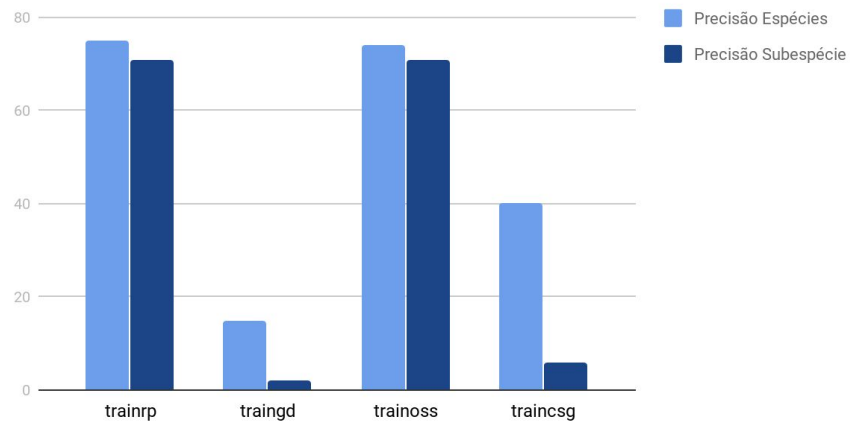
EX 1 PATTERNNET (FUNÇÃO DE ATIVAÇÃO) abordagem por parametros



12.4 Cascadeforwardnet

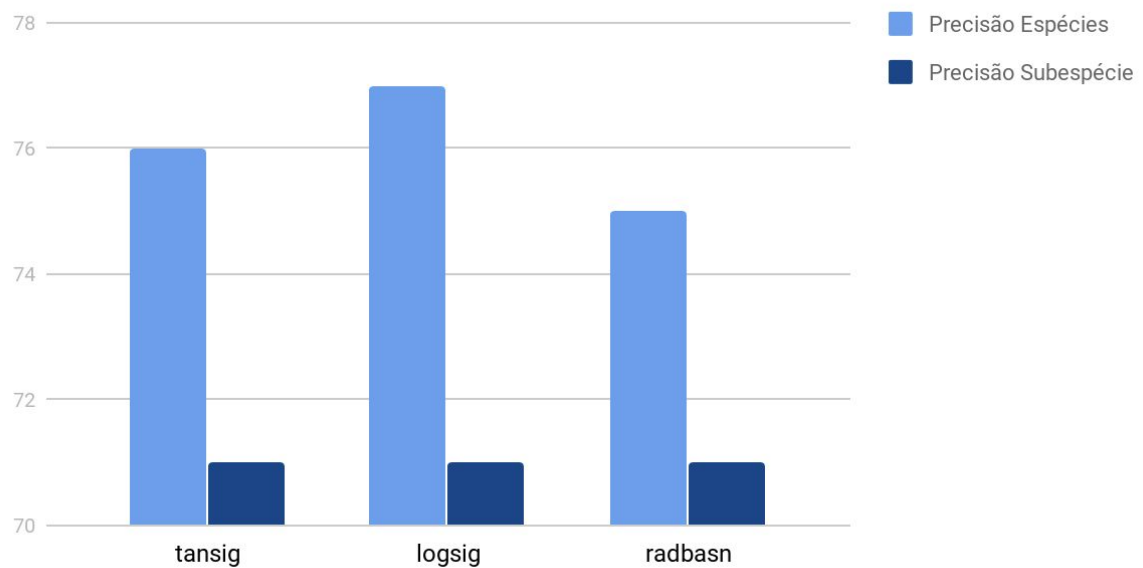
Funções de treino

EX1 CASCADEFORWARDNET (FUNÇÃO DE TREINO)
abordagem por parametros



Funções de ativação

EX 1 CASCADEFORWARDNET (FUNÇÃO DE ATIVAÇÃO)
abordagem por parametros



Conclusão

Mais uma vez podemos confirmar que o **input** de uma rede neuronal é o **principal factor** na escolha da sua **topologia** e **arquitetura**, usando o input original a rede não conseguiu obter resultados acima de 20%, e a pior das topologias era a **Cascadeforwardnet**, onde agora se nota que é sem dúvida a melhor solução com resultados perto dos 80%.

13- Menções Importantes

- A GUI nunca foi usado de um ponto de vista 100% de utilizador, pelo que pode apresentar alguns problemas não encontrados durante a criação da mesma.
- As redes neuronais foram gravadas usando uma sintaxe específica, pelo que para importar redes neuronais que não tenham sido criadas no mesmo programa requer uma atuação manual.
- As imagens são lidas usando o algoritmo de leitura de ficheiros do MATLAB, este não lê as imagens por sequência numérica mas por carácter, da mesma forma que uma base de dados procura caracteres, por isso a sequência das imagens não é por ID.
- Os códigos estão organizados por ordem alfabética da sua representação textual.
- As imagens carregadas manualmente não estão a ser tratadas de forma especial, pelo que seguem o mesmo princípio que o dataset, isto faz com que para uma leitura correcta estas devem ser manipuladas ou ajustadas antes.
- Deep Learning irá ajudar a ler imagens manuais e a melhorar a rede além do dataset inicial.
- Manipular imagens garantido que são plantas com uma rede neuronal só para isso seria uma abordagem que iria garantir resultados extremamente melhores.
- Manipular a orientação da planta de forma a garantir o caudal da folha sempre na mesma orientação iria garantir melhores resultados.
- Isolar os diversos componentes da folha e extrair informações de cada componente iria garantir melhores resultados.

14- Anexos

Juntamente com os gráficos disponibilizados ao longo deste trabalho assim como as diversas imagens em anexo existe um excel que contém todos os dados usados para estatísticas de dados.

15- Conclusão

Ao longo do trabalho chegamos a diversas conclusões, mas na perspectiva geral solidificamos o nosso conhecimento sobre redes neurais, principalmente como estruturar uma rede neuronal futuramente. Assim como no mundo real às vezes o problema não está nas respostas, mas sim nas perguntas, facilmente vimos que o simples facto do Input ter mudado drasticamente mudou os resultados obtidos, mesmo o target sendo o mesmo.

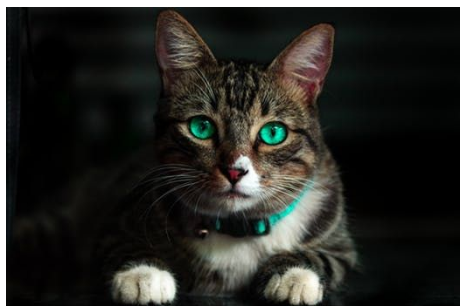
Quando resolvemos a alínea A perguntamo-nos:

Será que é proporcional ao tamanho do DataSet ou será da algoritmia de cada topologia/Função, será que o tipo de Input influencia drasticamente?”

Originalmente com poucos dados para responder a estas pergunta esperamos pelo fim, pois agora podemos dizer confortavelmente que, sim o DataSet usado inicialmente tem um peso duradouro na rede com parâmetros desconhecidos, o DataSet não é o principal factor na escolha de topologias e escolha de funções, mas é o principal factor em determinar a “inteligência” da rede em situações novas, quanto mais experiência mais equipada está para lidar com o desconhecido.

No que toca a redes neurais o principal alvo a atacar deve ser sempre a manipulação dos inputs, que input's devem ser considerados e como devem ser considerados. Como podem ser modificados e se podemos “simplificar” redes complexas em pequenas redes simples. Vamos considerar a seguinte analogia.

“Que cão é este?”



O ser humano facilmente consegue ver que não é um cão, mas uma rede neuronal não vai responder a isso, a rede neuronal vai determinar no melhor das suas possibilidades um resultado. De forma a garantirmos que uma rede é eficaz temos que garantir que os dados são igualmente bons, e é aí que a manipulação faz toda a diferença, e assim como vimos no nosso trabalho que ao mudarmos o nosso input também mudamos as “topologias” mais eficazes e aí está a ciência das redes neuronais, não no processamento mas sim qual o melhor processamento.