

# ENG1456 - Redes Neurais - Trabalho 1

## Classificação

Aluno: Matheus Carneiro Nogueira

Professora: Marley Velasco

### Sumário

<b>1</b>	<b>Apresentação do dataset</b>	<b>2</b>
<b>2</b>	<b>Compreensão do problema e análise de variáveis</b>	<b>2</b>
2.1	Observe a base de dados do problema. Existem variáveis que podem ser eliminadas do dataset? Justifique. . . . .	2
2.2	Implemente técnicas de visualização de dados e seleção de variáveis para extrair características importantes sobre a base de dados. Explique a motivação destas técnicas e o que é possível inferir dos resultados obtidos. . . .	3
<b>3</b>	<b>Treinamento do modelo de Rede Neural</b>	<b>3</b>
3.1	Com as configurações do modelo MLP previamente definidas no script, faça o treinamento da Rede Neural sem normalizar os atributos numéricos. Comente o resultado obtido, baseado nas métricas de avaliação disponíveis (acurácia, precision, recall, F1-Score, Matriz de confusão, etc.) . . . . .	3
3.2	Agora normalize os dados de entrada e treine novamente o modelo MLP. Avalie os resultados obtidos e comente o efeito da normalização no treinamento da Rede Neural. . . . .	6
<b>4</b>	<b>Mudança de configurações do modelo</b>	<b>8</b>
4.1	Insira o conjunto de validação para o treinamento do modelo. Avalie o resultado obtido. . . . .	8
4.2	Modifique o tempo de treinamento (épocas) da Rede Neural. Escolha dois valores distintos (e.g. 1 e 1000 épocas) e avalie os resultados. . . . .	8
4.3	Modifique a taxa de aprendizado da Rede Neural. Escolha dois valores distintos (e.g. 0,001 e 0,1) e avalie os resultados. . . . .	8
4.4	Modifique a quantidade de neurônios na camada escondida da Rede Neural. Escolha dois valores distintos (e.g. 2 e 70 neurônios) e avalie os resultados. . . . .	8
<b>5</b>	<b>Teste Livre</b>	<b>8</b>
5.1	Faça novos testes para avaliar o desempenho da Rede Neural no problema designado. Use a técnica K-Fold (com $K = 10$ ) para analisar o resultado obtido. . . . .	8
5.2	Faça análises e novas implementações que você julgue importante para o seu trabalho. Não esqueça de explicar a motivação da análise realizada. . .	8

## Resumo

Este documento consiste no relatório do trabalho 1 do módulo de Redes Neurais da disciplina ENG1456 da PUC-Rio. Nele será explicada a implementação de modelos de Redes Neurais MLP para a classificação do dataset Ionosphere, disponibilizado pela professora da disciplina e disponível em [1]. As seções do relatório são definidas de acordo com as perguntas principais que constam no arquivo Guia de Atividades.

## 1 Apresentação do dataset

Com o intuito de criar um modelo de rede neural para classificar o dataset *Ionosphere*, é essencial que, antes de implementar a rede, estudemos o dataset em si. Como descrito no artigo [2] e pelas informações disponibilizadas em [1], este dataset consiste no sistema de radares da região de Goose Bay, Labrador, o qual possui 16 antenas de alta frequência. O alvo dessas antenas foram elétrons livres na Ionosfera e, bons retornos do radar consistem em retornos que indicam alguma estrutura na ionosfera, enquanto más retornos são aqueles cujo sinal passa livre pela Ionosfera e não retorna. É, justamente, esta classificação que a Rede Neural desenvolvida neste trabalho almeja realizar, separar as entradas em "good" ou "bad". Com isso, notamos que a rede pode possuir uma saída única, que mapeia 0 em "bad" e 1 em "good".

## 2 Compreensão do problema e análise de variáveis

### 2.1 Observe a base de dados do problema. Existem variáveis que podem ser eliminadas do dataset? Justifique.

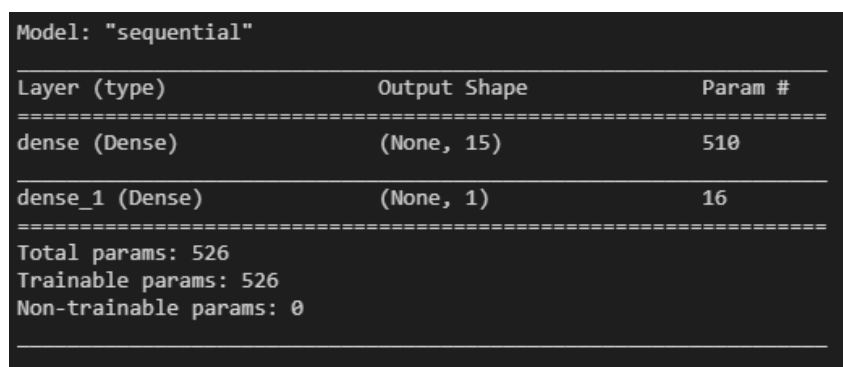
Ao analisar as colunas do dataset, percebe-se que as duas primeiras colunas, *info\_1* e *info\_2* aparentam ser colunas inteiramente de 1 e 0, respectivamente. Para ter certeza, uma vez que observar apenas o head e tail não é suficiente, as colunas inteiras foram analisadas e confirmou-se que a segunda coluna, *info\_1* de fato é inteiramente de 0. Sendo assim, é recomendável que ela seja excluída do dataset pois, além de não trazer nenhuma informação útil à classificação dos retornos de rádio, essa informação pode atrapalhar o treinamento da rede, uma vez que ambos os retornos bons e ruins apresentam valor 0 neste atributo.

**2.2** Implemente técnicas de visualização de dados e seleção de variáveis para extrair características importantes sobre a base de dados. Explique a motivação destas técnicas e o que é possível inferir dos resultados obtidos.

### 3 Treinamento do modelo de Rede Neural

**3.1** Com as configurações do modelo MLP previamente definidas no script, faça o treinamento da Rede Neural sem normalizar os atributos numéricos. Comente o resultado obtido, baseado nas métricas de avaliação disponíveis (acurácia, precision, recall, F1-Score, Matriz de confusão, etc.)

O nosso primeiro modelo de rede MLP foi criado de acordo com as especificações já presentes no script disponibilizado. Essa Rede Neural possui as seguintes especificações:



```
Model: "sequential"
Layer (type)                 Output Shape              Param #
-----
dense (Dense)                 (None, 15)                510
dense_1 (Dense)               (None, 1)                 16
-----
Total params: 526
Trainable params: 526
Non-trainable params: 0
```

Figura 1: Sumário do Modelo MLP Não Normalizado

Como a figura 1 revela, nossa rede possui 1 camada escondida completamente conectada (densa) com 15 neurônios e a camada de saída com apenas 1 neurônio, cujo motivo fora explicado na seção 1. Além disso, nota-se que o número de parâmetros treináveis é 526. Essa informação será importante quando variarmos o número de épocas de treinamento, que neste caso é 50, uma vez que possuir épocas demais pode significar um overfitting de nosso modelo, gerando perda de generalização.

Para analisar a acurácia deste modelo, observemos as figuras abaixo.

```

Epoch 1/50
9/9 [=====] - 1s 4ms/step - loss: 0.5498 - accuracy: 0.7346
Epoch 2/50
9/9 [=====] - 0s 1ms/step - loss: 0.2430 - accuracy: 0.9342
Epoch 3/50
9/9 [=====] - 0s 1ms/step - loss: 0.2149 - accuracy: 0.9181
Epoch 4/50
9/9 [=====] - 0s 1ms/step - loss: 0.1356 - accuracy: 0.9420
Epoch 5/50
9/9 [=====] - 0s 1ms/step - loss: 0.1175 - accuracy: 0.9635
Epoch 6/50
9/9 [=====] - 0s 1ms/step - loss: 0.1242 - accuracy: 0.9543
Epoch 7/50
9/9 [=====] - 0s 1ms/step - loss: 0.0903 - accuracy: 0.9630
Epoch 8/50
9/9 [=====] - 0s 1ms/step - loss: 0.0681 - accuracy: 0.9764
Epoch 9/50
9/9 [=====] - 0s 1ms/step - loss: 0.0825 - accuracy: 0.9806
Epoch 10/50
9/9 [=====] - 0s 1ms/step - loss: 0.0431 - accuracy: 0.9858
Epoch 11/50
9/9 [=====] - 0s 1ms/step - loss: 0.0483 - accuracy: 0.9829
Epoch 12/50
9/9 [=====] - 0s 1ms/step - loss: 0.0428 - accuracy: 0.9937
Epoch 13/50
9/9 [=====] - 0s 1ms/step - loss: 0.0451 - accuracy: 0.9838
Epoch 14/50
9/9 [=====] - 0s 983us/step - loss: 0.0339 - accuracy: 0.9948
Epoch 15/50
9/9 [=====] - 0s 951us/step - loss: 0.0500 - accuracy: 0.9899
Epoch 16/50
9/9 [=====] - 0s 1ms/step - loss: 0.0255 - accuracy: 0.9954
Epoch 17/50
9/9 [=====] - 0s 1ms/step - loss: 0.0245 - accuracy: 0.9973
Epoch 18/50
9/9 [=====] - 0s 1ms/step - loss: 0.0283 - accuracy: 0.9938
Epoch 19/50
9/9 [=====] - 0s 1ms/step - loss: 0.0277 - accuracy: 0.9965
Epoch 20/50
9/9 [=====] - 0s 1ms/step - loss: 0.0475 - accuracy: 0.9908
Epoch 21/50
9/9 [=====] - 0s 1ms/step - loss: 0.0504 - accuracy: 0.9908
Epoch 22/50
9/9 [=====] - 0s 1ms/step - loss: 0.0439 - accuracy: 0.9863
Epoch 23/50
9/9 [=====] - 0s 922us/step - loss: 0.0208 - accuracy: 0.9950
Epoch 24/50
9/9 [=====] - 0s 1ms/step - loss: 0.0465 - accuracy: 0.9908
Epoch 25/50
9/9 [=====] - 0s 1ms/step - loss: 0.0243 - accuracy: 0.9939
Epoch 26/50
9/9 [=====] - 0s 1ms/step - loss: 0.0188 - accuracy: 0.9978
Epoch 27/50
9/9 [=====] - 0s 1ms/step - loss: 0.0176 - accuracy: 0.9973
Epoch 28/50
9/9 [=====] - 0s 1ms/step - loss: 0.0116 - accuracy: 0.9993
Epoch 29/50
9/9 [=====] - 0s 989us/step - loss: 0.0180 - accuracy: 0.9965
Epoch 30/50
9/9 [=====] - 0s 1ms/step - loss: 0.0111 - accuracy: 0.9993
Epoch 31/50
9/9 [=====] - 0s 1ms/step - loss: 0.0135 - accuracy: 0.9989
Epoch 32/50
9/9 [=====] - 0s 1ms/step - loss: 0.0088 - accuracy: 0.9989
Epoch 33/50
9/9 [=====] - 0s 987us/step - loss: 0.0182 - accuracy: 0.9955
Epoch 34/50
9/9 [=====] - 0s 1ms/step - loss: 0.0176 - accuracy: 0.9944
Epoch 35/50
9/9 [=====] - 0s 1ms/step - loss: 0.0193 - accuracy: 0.9955
Epoch 36/50
9/9 [=====] - 0s 1ms/step - loss: 0.0123 - accuracy: 0.9984
Epoch 37/50
9/9 [=====] - 0s 1ms/step - loss: 0.0187 - accuracy: 0.9939
Epoch 38/50
9/9 [=====] - 0s 1ms/step - loss: 0.0127 - accuracy: 0.9984
Epoch 39/50
9/9 [=====] - 0s 1ms/step - loss: 0.0061 - accuracy: 0.9979
Epoch 40/50
9/9 [=====] - 0s 2ms/step - loss: 0.0094 - accuracy: 0.9984
Epoch 41/50
9/9 [=====] - 0s 1ms/step - loss: 0.0146 - accuracy: 0.9908
Epoch 42/50
9/9 [=====] - 0s 1ms/step - loss: 0.0087 - accuracy: 0.9973
Epoch 43/50
9/9 [=====] - 0s 1ms/step - loss: 0.0136 - accuracy: 0.9908
Epoch 44/50
9/9 [=====] - 0s 1ms/step - loss: 0.0103 - accuracy: 0.9939
Epoch 45/50
9/9 [=====] - 0s 1ms/step - loss: 0.0044 - accuracy: 0.9979
Epoch 46/50
9/9 [=====] - 0s 988us/step - loss: 0.0063 - accuracy: 0.9993
Epoch 47/50
9/9 [=====] - 0s 1ms/step - loss: 0.0245 - accuracy: 0.9908
Epoch 48/50
9/9 [=====] - 0s 1ms/step - loss: 0.0053 - accuracy: 0.9965
Epoch 49/50
9/9 [=====] - 0s 1ms/step - loss: 0.0051 - accuracy: 0.9965
Epoch 50/50
9/9 [=====] - 0s 1ms/step - loss: 0.0036 - accuracy: 1.0000

```

Figura 2: Output do treinamento do modelo não normalizado

Confusion Matrix				
[[23 5]				
[ 1 42]]				
Classification Report				
	precision	recall	f1-score	support
0	0.96	0.82	0.88	28
1	0.89	0.98	0.93	43
accuracy			0.92	71
macro avg	0.93	0.90	0.91	71
weighted avg	0.92	0.92	0.91	71

Figura 3: Métricas do modelo não normalizado

O modelo em questão apresentou acurácia de 92%, o que, a priori, é bom, vide figura 3. Essa métrica indica que em 92% dos casos apresentados na fase de validação a rede foi capaz de classificar corretamente o retorno como bom ou ruim. Podemos observar, também, que a evolução da acurácia do modelo na figura 2. Percebe-se que, uma vez atingida a acurácia de 0.99 houve pouca flutuação nos valores dessa métrica, que na última época atingiu o valor de 1.00. Isso pode indicar que a quantidade de épocas não foi grande demais, pois, se fosse, veríamos uma queda maior na acurácia depois de certa

época ou flutuações maiores em seu valor. Novamente, apenas quando testarmos outros valores de época poderemos confirmar se este foi (50) é um bom valor, mas, por hora, ele parece razoável.

A precisão de nosso modelo foi de 0.96 para os casos zero, isto é, "bad" e de 0.89 para 1, "good". Isso nos mostra que a rede possui um grande desempenho em classificar corretamente os retornos ruins e um desempenho um pouco pior em classificar os retornos bons. Isso é curioso uma vez que a nossa base de treino foi composta de 182 entradas 1 e 98 entradas 0, o que mostra que nossos dados são razoavelmente desbalanceados. Uma solução possível para melhorar essa métrica é construir a base de treino de outra forma, garantindo maior equilíbrio entre os retornos bons e ruins. Na seção ?? faremos esse teste. Para a situação deste dataset, podemos argumentar que é melhor a rede classificar um retorno bom como ruim do que um retorno ruim como bom, uma vez que este segundo erro poderia causar a utilização de uma antena defeituosa, o que é mais prejudicial que não utilizar uma antena em bom funcionamento. Sendo assim, a rede indicar uma precisão maior para os retornos ruins é melhor do que o caso contrário, se ela apresentasse precisão menor para retornos ruins.

A métrica recall indica quanto a rede acertou a classificação de uma entrada da classe 0, por exemplo, em relação ao número de entradas da classe 0. É justamente o recall que nós podemos observar na matriz de confusão da figura 3. Nosso resultado indica que, em 82% dos casos em que o retorno era ruim, nosso modelo acertou que ele era ruim e, em 98% dos casos em que o retorno era bom, a rede classificou corretamente que era bom. Esta métrica usada em conjunto com a precisão é útil para avaliar o quão preciso é nosso modelo em classificar corretamente as entradas em relação à classe correta da entrada. Como nossa precisão é maior para a classe "ruim" e o recall é menor para essa mesma classe, podemos ter maior confiança do que se ambas as métricas fossem maiores para uma classe específica, o que indicaria que nosso modelo não é muito bom em classificar a entrada nesta classe.

Por fim, o F1-score indica a média harmônica entre a Precisão e o Recall. **Completar, não entendi muito bem!!**

A figura abaixo indica a matriz de confusão relativa de nosso modelo. Como nós já comentamos sobre ela e sobre a métrica Recall, aqui apenas consta um resumo de sua interpretação.

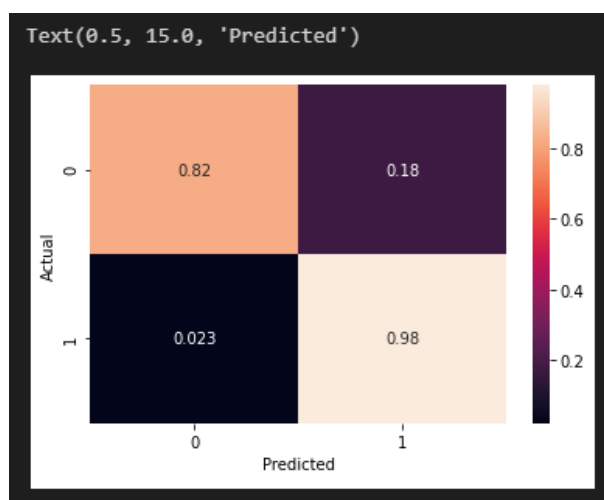


Figura 4: 82% das entradas ruins foram classificadas como ruins e 98% das boas foram classificadas como boas

### 3.2 Agora normalize os dados de entrada e treine novamente o modelo MLP. Avalie os resultados obtidos e comente o efeito da normalização no treinamento da Rede Neural.

Realizar uma normalização dos dados de entrada é uma estratégia útil para tentar aumentar tanto a eficiência computacional quanto sua precisão de classificação. A normalização é importante, também, para adequar os valores aos domínios das funções de ativação que, neste caso, são a *ReLU* (Rectified Linear Activation Unit) utilizada para as camadas escondidas e a *Sigmoid* para a camada de saída.

Ao normalizar as entradas, novamente foi verificada a coluna *info\_0*, que permaneceu na base de dados na seção 2.1. Ela permanece, obviamente, inalterada, uma vez que seus valores já eram em 0 e 1. No entanto, foi modelada uma rede com e outra sem esta coluna e não foi observada grande diferença de qualidade de classificação. Sendo assim, a coluna permanece na base de dados.

Além de normalizar, nenhuma outra modificação foi feita no modelo, então podemos seguir para a explanação dos resultados. As figuras a seguir exibem o output do treinamento do modelo e as métricas dos resultados de teste.

```

Epoch 1/50
9/9 [=====] - 0s 1ms/step - loss: 0.7356 - accuracy: 0.4884
Epoch 2/50
9/9 [=====] - 0s 1ms/step - loss: 0.7056 - accuracy: 0.6358
Epoch 3/50
9/9 [=====] - 0s 1ms/step - loss: 0.5348 - accuracy: 0.7287
Epoch 4/50
9/9 [=====] - 0s 970us/step - loss: 0.5210 - accuracy: 0.7662
Epoch 5/50
9/9 [=====] - 0s 1ms/step - loss: 0.5084 - accuracy: 0.7749
Epoch 6/50
9/9 [=====] - 0s 975us/step - loss: 0.4219 - accuracy: 0.8317
Epoch 7/50
9/9 [=====] - 0s 972us/step - loss: 0.3722 - accuracy: 0.8612
Epoch 8/50
9/9 [=====] - 0s 1ms/step - loss: 0.3525 - accuracy: 0.8647
Epoch 9/50
9/9 [=====] - 0s 1ms/step - loss: 0.3508 - accuracy: 0.8537
Epoch 10/50
9/9 [=====] - 0s 970us/step - loss: 0.3025 - accuracy: 0.9011
Epoch 11/50
9/9 [=====] - 0s 1ms/step - loss: 0.3187 - accuracy: 0.8750
Epoch 12/50
9/9 [=====] - 0s 1ms/step - loss: 0.2713 - accuracy: 0.9133
Epoch 13/50
9/9 [=====] - 0s 964us/step - loss: 0.2835 - accuracy: 0.8950
Epoch 14/50
9/9 [=====] - 0s 986us/step - loss: 0.2495 - accuracy: 0.9040
Epoch 15/50
9/9 [=====] - 0s 1ms/step - loss: 0.3090 - accuracy: 0.8594
Epoch 16/50
9/9 [=====] - 0s 1ms/step - loss: 0.2461 - accuracy: 0.9203
Epoch 17/50
9/9 [=====] - 0s 1ms/step - loss: 0.2091 - accuracy: 0.9373
Epoch 18/50
9/9 [=====] - 0s 2ms/step - loss: 0.2109 - accuracy: 0.9247
Epoch 19/50
9/9 [=====] - 0s 1ms/step - loss: 0.1862 - accuracy: 0.9386
Epoch 20/50
9/9 [=====] - 0s 1ms/step - loss: 0.1737 - accuracy: 0.9545
Epoch 21/50
9/9 [=====] - 0s 1ms/step - loss: 0.1542 - accuracy: 0.9391
Epoch 22/50
9/9 [=====] - 0s 1ms/step - loss: 0.1859 - accuracy: 0.9269
Epoch 23/50
9/9 [=====] - 0s 1ms/step - loss: 0.1783 - accuracy: 0.9347
Epoch 24/50
9/9 [=====] - 0s 4ms/step - loss: 0.2664 - accuracy: 0.8831
Epoch 25/50
9/9 [=====] - 0s 2ms/step - loss: 0.2407 - accuracy: 0.9104
Epoch 26/50
9/9 [=====] - 0s 1ms/step - loss: 0.1635 - accuracy: 0.9470
Epoch 27/50
9/9 [=====] - 0s 1ms/step - loss: 0.2162 - accuracy: 0.9136
Epoch 28/50
9/9 [=====] - 0s 1ms/step - loss: 0.1453 - accuracy: 0.9339
Epoch 29/50
9/9 [=====] - 0s 1ms/step - loss: 0.1772 - accuracy: 0.9435
Epoch 30/50
9/9 [=====] - 0s 1ms/step - loss: 0.2311 - accuracy: 0.9037
Epoch 31/50
9/9 [=====] - 0s 1ms/step - loss: 0.1285 - accuracy: 0.9471
Epoch 32/50
9/9 [=====] - 0s 1ms/step - loss: 0.1284 - accuracy: 0.9514
Epoch 33/50
9/9 [=====] - 0s 1ms/step - loss: 0.1558 - accuracy: 0.9508
Epoch 34/50
9/9 [=====] - 0s 2ms/step - loss: 0.1108 - accuracy: 0.9640
Epoch 35/50
9/9 [=====] - 0s 1ms/step - loss: 0.1461 - accuracy: 0.9420
Epoch 36/50
9/9 [=====] - 0s 1ms/step - loss: 0.1507 - accuracy: 0.9438
Epoch 37/50
9/9 [=====] - 0s 1ms/step - loss: 0.1401 - accuracy: 0.9533
Epoch 38/50
9/9 [=====] - 0s 1ms/step - loss: 0.1820 - accuracy: 0.9333
Epoch 39/50
9/9 [=====] - 0s 1ms/step - loss: 0.1384 - accuracy: 0.9279
Epoch 40/50
9/9 [=====] - 0s 1ms/step - loss: 0.1469 - accuracy: 0.9379
Epoch 41/50
9/9 [=====] - 0s 1ms/step - loss: 0.1300 - accuracy: 0.9604
Epoch 42/50
9/9 [=====] - 0s 1ms/step - loss: 0.1415 - accuracy: 0.9411
Epoch 43/50
9/9 [=====] - 0s 1ms/step - loss: 0.1242 - accuracy: 0.9660
Epoch 44/50
9/9 [=====] - 0s 1ms/step - loss: 0.1092 - accuracy: 0.9673
Epoch 45/50
9/9 [=====] - 0s 1ms/step - loss: 0.1104 - accuracy: 0.9682
Epoch 46/50
9/9 [=====] - 0s 1ms/step - loss: 0.0921 - accuracy: 0.9826
Epoch 47/50
9/9 [=====] - 0s 1ms/step - loss: 0.1431 - accuracy: 0.9441
Epoch 48/50
9/9 [=====] - 0s 1ms/step - loss: 0.1508 - accuracy: 0.9454
Epoch 49/50
9/9 [=====] - 0s 1ms/step - loss: 0.1170 - accuracy: 0.9711
Epoch 50/50
9/9 [=====] - 0s 1ms/step - loss: 0.1161 - accuracy: 0.9646

```

Figura 5: Output do treinamento do modelo normalizado

Confusion Matrix					
[[24 4]					
[ 0 43]]					
Classification Report					
	precision	recall	f1-score	support	
0	1.00	0.86	0.92	28	
1	0.91	1.00	0.96	43	
accuracy			0.94	71	
macro avg	0.96	0.93	0.94	71	
weighted avg	0.95	0.94	0.94	71	

Figura 6: Métricas do modelo normalizado

A acurácia do modelo normalizado é maior que a do modelo não normalizado. Obtivemos um aumento de 0.02 nesta métrica, passando de 0.92 para 0.94. Como nossa acurácia já possuía um valor alto, este aumento, por si só, pode não significar grandes melhoras gerais da qualidade da rede, então é importante analisarmos, novamente, as demais métricas. Além disso, ao analisar a evolução da acurácia na figura 5, percebe-se que houve maiores flutuações do valor da acurácia e que foram necessárias mais épocas para ela atingir pela primeira vez um valor acima de 0.9. No entanto, como todos os pesos são inicializados aleatoriamente, este fato não é de grande importância para averiguar a qualidade da rede.

Ambas as precisões das classes "ruim" e "boa" aumentaram. A precisão da classe 0 (ruim) aumentou de 0.96 para 1.0 e a classe 1 (boa) aumentou de 0.89 para 0.91. Isso quer dizer que a rede cujas entradas foram normalizadas se mostrou a rede não classificou nenhuma entrada ruim como boa e classificou apenas 0.09 entradas boas como ruins. Esse fato aumenta a confiabilidade desta rede em relação à anterior.

O recall, de modo similar à precisão, também aumentou seus valores. Para a classe "ruim" o modelo apresentou uma evolução de 0.82 para 0.86 e, para a classe "boa", de 0.98 para 1. Novamente, isso quer dizer que este modelo

## 4 Mudança de configurações do modelo

- 4.1 Insira o conjunto de validação para o treinamento do modelo. Avalie o resultado obtido.
- 4.2 Modifique o tempo de treinamento (épocas) da Rede Neural. Escolha dois valores distintos (e.g. 1 e 1000 épocas) e avalie os resultados.
- 4.3 Modifique a taxa de aprendizado da Rede Neural. Escolha dois valores distintos (e.g. 0,001 e 0,1) e avalie os resultados.
- 4.4 Modifique a quantidade de neurônios na camada escondida da Rede Neural. Escolha dois valores distintos (e.g. 2 e 70 neurônios) e avalie os resultados.

## 5 Teste Livre

- 5.1 Faça novos testes para avaliar o desempenho da Rede Neural no problema designado. Use a técnica K-Fold (com  $K = 10$ ) para analisar o resultado obtido.
- 5.2 Faça análises e novas implementações que você julgue importante para o seu trabalho. Não esqueça de explicar a motivação da análise realizada.

## Referências

- [1] UCI Machine Learning Repository,  
<https://archive.ics.uci.edu/ml/datasets/Ionosphere>
- [2] Sigillito, V. G., Wing, S. P., Hutton, L. V., & Baker, K. B. (1989). *Classification of radar returns from the ionosphere using neural networks*. Johns Hopkins APL Technical Digest, 10, 262-266.