UNIVERSIDADE ESTADUAL PAULISTA
Programa de Pós-Graduação em Ciência da Computação
Computação Inspirada pela Natureza - **Trabalho 2** - Professor Fabricio Breve
Giovanna Carreira Marinho

Introdução

Inspiradas no funcionamento do sistema nervoso, as Redes Neurais Artificiais (RNA) constituem uma ferramenta poderosa para diferentes tarefas: classificação, regressão, entre outras. Por meio de uma arquitetura composta por nós (representando artificialmente os neurônios) conectados a partir de um padrão (definindo a estrutura ou arquitetura da rede) o método de treinamento determina os pesos da rede e permite o aprendizado de algum padrão. Ou seja, o aprendizado da rede consiste em encontrar as conexões apropriadas para produzir um padrão de ativação satisfatório sob certas circunstâncias. Uma vez treinada a rede, ela estará apta a gerar respostas para novos dados.

Nas próximas duas seções a seguir, são apresentados dois exemplos de aplicação de uma Rede Neural Artificial, mais precisamente uma rede *Perceptron* treinada com algoritmo *Backpropagation*, para reconhecimento de classes para dois conjuntos de dados: *Iris* (três espécies de flores) e *Wine* (três tipos de vinhos).

A rede *Perceptron* utilizada nesses exemplos possui uma camada de entrada (com número de neurônios igual a quantidade de atributos), uma camada oculta (com número de neurônios igual a média geométrica entre a quantidade de neurônios da camada de entrada e da camada de saída) e uma camada de saída (com número de neurônios igual a quantidade de classes do problema). Todos os neurônios são conectados com os neurônios da camada anterior e os pesos iniciais são atribuídos aleatoriamente em um dado intervalo. Essa inicialização aleatória é importante pois caso esse processo fosse feito com um único valor padrão (como zero, por exemplo) o treinamento ia falhar, já que todos os neurônios seriam perfeitamente idênticos e o processo de retro propagação (*Backpropagation*) iria “afetar” os pesos a mesma maneira e eles iriam permanecer iguais. Isso iria funcionar como se a rede tivesse um neurônio por camada, ao invés de vários. Com essa inicialização aleatória, a simetria é quebrada e a retro propagação permite treinar uma quantidade diversificada de neurônios. A função de ativação/transferência utilizada na camada oculta e de saída é a logística, que tem uma derivada diferente de zero bem definida em todos os lugares, permitindo de o gradiente descendente tenha algum progresso em cada etapa.

Como dito anteriormente o treinamento da rede é feito pelo algoritmo *Backpropagation*. Ao final de cada época, é feito o cálculo do erro da rede por meio da seguinte fórmula:

$$E_r = \frac{1}{2}\sum_{k=1}^{M}(\delta_k^s)^2$$

sendo M a quantidade de neurônios na camada de saída e δ_k^s o respectivo erro nessa camada (calculado pela diferença do valor desejado e obtido multiplicado pela derivada da função de transferência da camada de saída).

Os passos necessários para o treinamento da rede são feitos até que uma quantidade de iteração máxima seja alcançada ou que o erro daquela época atinja um erro mínimo que também pode ser passado como parâmetro da função de treinamento. Se esses dois valores forem definidos (quantidade máxima de iterações e erro mínimo da rede), o treinamento para quando o primeiro for atingido.

Durante o processo de treinamento, o erro quadrado médio (MSE) é calculado para o conjunto de treinamento e validação. Para o processo de classificação de dados, feito após o treinamento completo da rede, o conjunto de teste é utilizado. Nesse caso é calculada a matriz de confusão e o erro quadrado médio (MSE).

As próximas duas seções apresentam a arquitetura da rede para cada tipo de problema e os testes realizados para treinamento e classificação.

A última seção apresenta dois exemplos de classificação de imagens. O primeiro exemplo também utiliza uma rede *Perceptron*, entretanto outras camadas são utilizadas para cobrir o escopo do problema: classificar imagens em tons de cinza de itens de roupas. O segundo exemplo utilizada uma Rede Neural Convolucional para classificação de imagens coloridas de flores. Para ambos os exemplos é utilizado o pacote *Keras* para a criação das redes neurais.

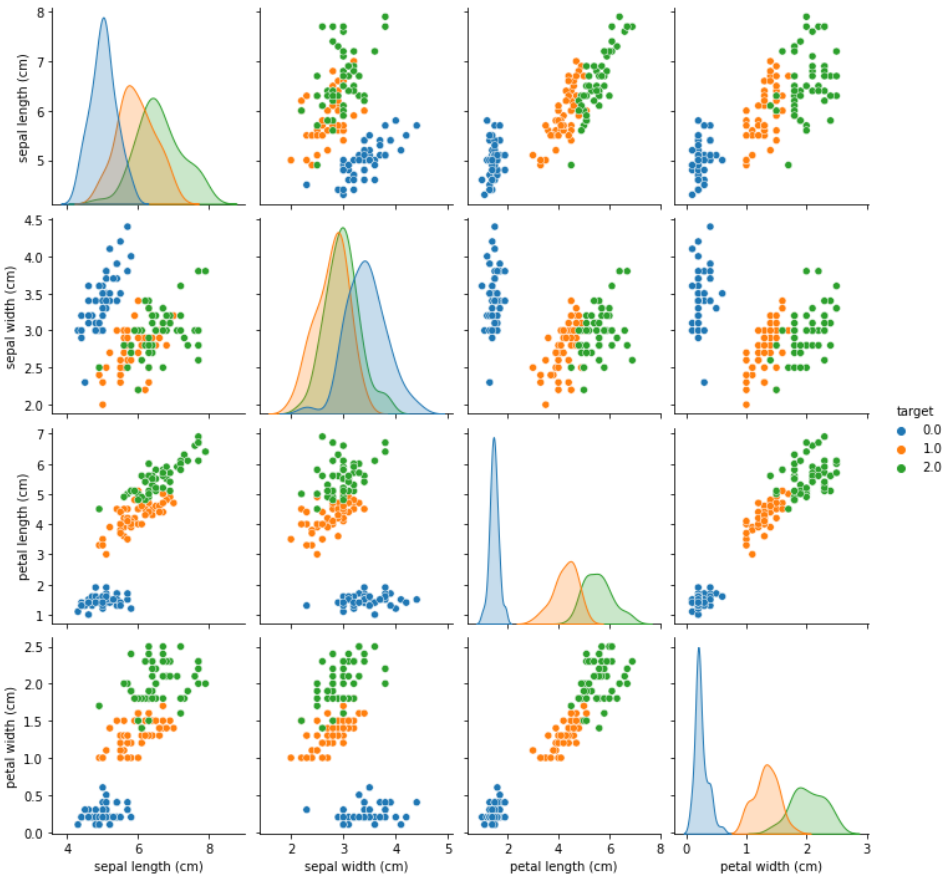
Para o desenvolvimento dos algoritmos, foi utilizada a linguagem de programação Python e o ambiente de desenvolvimento Google Colab. As implementações podem ser acessadas pelo link: https://github.com/Giovannacm/nature-inspired-computing/blob/main/RNA.ipynb.

Conjunto de dados *Iris*

O conjunto de dados *Iris* consiste em 150 instâncias de dados de 3 classes da flor *Iris*: *setosa*, *versicolor* e *virginica*, sendo 50 de cada classe. Cada instância possui informações de quatro atributos: comprimento da sépala, largura da sépala, comprimento da pétala e largura da pétala. A Figura a seguir apresenta a relação entre esses atributos e classes.

Dessa forma, seguindo a definição da quantidade de neurônios apresentada anteriormente: a camada de entrada apresenta 4 neurônios; a camada oculta, 3 neurônios; a camada de saída, 3 neurônios.

Os dados foram divididos na proporção: 70% treinamento, 15% validação e 15% teste.

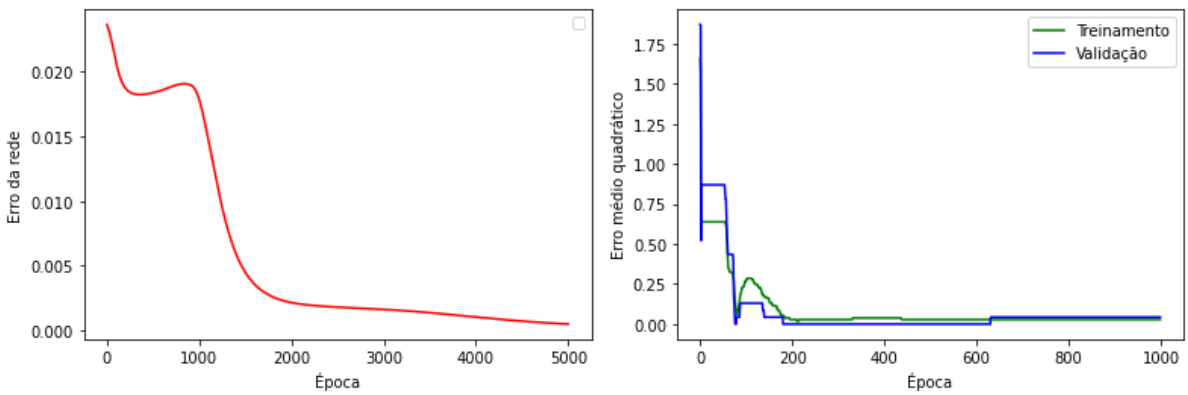


Os testes realizados com a rede são apresentados a seguir. Diferentes taxas de aprendizado são exploradas.

Inicialmente o processo de treinamento foi executado apenas uma vez, para analisar um comportamento inicial da rede e do problema. Executando o processo de treinamento com taxa de aprendizado 0.01 por 1000 épocas e com pesos iniciais aleatórios entre -0.1 e 0.1, obteve-se na última época:

- Erro da rede: 0.00016
- MSE treinamento: 0.028571
- MSE validação: 0.043478

Exibindo o comportamento desses valores durante as épocas temos:



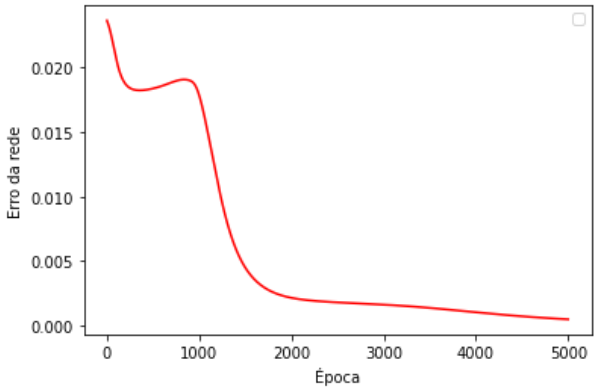
Realizando o processo de classificação para os dados de teste, ou seja, novos dados que não foram apresentados para a rede, observa-se que o MSE foi de 0.090909 a matriz de confusão de teste ficou configurada como:

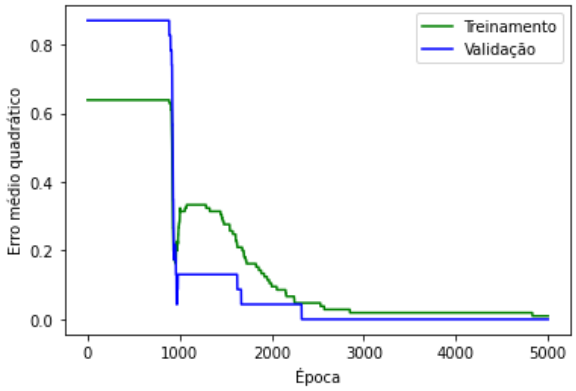
	Desejado		
Obtido	**Classe 1**	**Classe 2**	**Classe 3**
Classe 1	7	0	0
Classe 2	0	7	0
Classe 3	0	2	6

Ou seja, 2 amostras que eram da classe 2 foram classificadas como sendo da classe 3, fato que explica o MSE obtido. Analisando a taxa de erro e acerto, observa-se que a taxa de acerto foi de 90.91% e a de erro de 9.09%.

Alterando a taxa de aprendizado para 0.001, observa-se que para obter um resultado melhor é necessária uma quantidade maior de iterações (5000), já que a rede demora mais para convergir. Nessa configuração os valores obtidos na última iteração foram:

- Erro da rede: 0.000502
- MSE treinamento: 0.009524
- MSE validação: 0.0





O processo de classificação do conjunto de teste para essa rede, obteve um MSE de 0.045454 e a matriz de confusão abaixo:

	Desejado		
Obtido	**Classe 1**	**Classe 2**	**Classe 3**
Classe 1	7	0	0
Classe 2	0	8	0
Classe 3	0	1	6

Nesse caso, foi classificada errada apenas 1 amostra que era da classe 2 e foi classificada como sendo da classe 3, fato que explica o MSE obtido (menor que o anterior). A taxa de acerto foi de 95.45% e a de erro 4.54%. Observando a matriz de confusão para o conjunto de treinamento, temos:

	Desejado		
Obtido	**Classe 1**	**Classe 2**	**Classe 3**
Classe 1	33	0	0
Classe 2	0	37	0
Classe 3	0	1	34

O MSE do treinamento, já apresentado anteriormente, foi de 0.009524. A taxa de acerto, nesse caso foi de 99.05% e a de erro de 0.95%. Por fim, analisando os mesmos valores para o conjunto de validação, temos a seguinte matriz de confusão:

	Desejado		
Obtido	**Classe 1**	**Classe 2**	**Classe 3**
Classe 1	10	0	0
Classe 2	0	3	0
Classe 3	0	0	10

O conjunto de validação obteve um MSE de 0, uma vez que a taxa de acerto foi de 100%, já que todas as classes foram classificadas corretamente.

O exemplo a seguir executa a mesma configuração da rede 30 vezes, testando diferentes taxas de aprendizado. A rede é configurada para ter pesos aleatórios entre -0.1 e 0.1 e 700 épocas de treinamento. Observa-se ao final de cada lote de 30 execuções o erro da rede, a média do MSE de treinamento e validação da última época, além do MSE de teste.

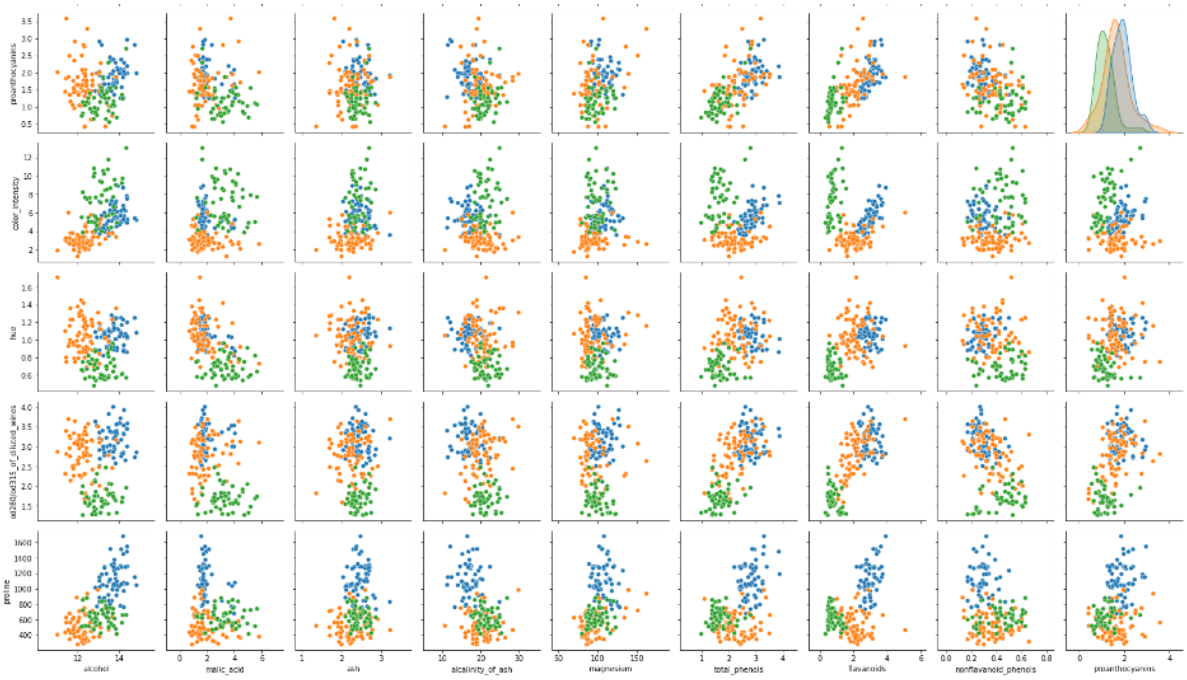
```
Taxa de aprendizado: 0.0001
 -> Erro da rede (avg): 0.021693
 -> MSE treinamento (avg): 0.902222
 -> MSE validação (avg): 1.049275
 -> MSE teste (avg): 0.960606

Taxa de aprendizado: 0.001
 -> Erro da rede (avg): 0.016113
 -> MSE treinamento (avg): 0.401905
 -> MSE validação (avg): 0.468116
 -> MSE teste (avg): 0.419697

Taxa de aprendizado: 0.01
 -> Erro da rede (avg): 0.000319
 -> MSE treinamento (avg): 0.050159
 -> MSE validação (avg): 0.026087
 -> MSE teste (avg): 0.063636
```

Os valores evidenciam que quanto menor a taxa de aprendizado, maior deve ser a quantidade de épocas para treinar a rede. Com uma taxa de aprendizado maior (0.01), observa-se que, após 30 execuções, o MSE médio de treinamento, validação e teste foi

consideravelmente menor que aqueles obtidos com as taxas anteriores, indicando uma convergência mais rápida nessa situação. Com a taxa de aprendizado de 0.0001 a convergência do algoritmo seria muito tardia, nesse caso seria necessária mais épocas de treinamento para que se obtivesse melhores resultados.

Conjunto de dados *Wine*

Já o conjunto de dados *Wine* consiste em 178 instâncias de dados de 3 tipos de vinhos (59 para a classe 0, 71 para a classe 1 e 48 para a classe 2). Cada instância possui informações de 13 atributos: álcool, ácido málico, cinzas, alcalinidade das cinzas, magnésio, fenóis totais, flavonoides, fenóis não flavonoides, proantocianinas, intensidade de cor, matiz, OD280/OD315 de vinhos diluídos e prolina. Uma vez carregados os dados, eles foram normalizados pelo método *zscore*. A Figura a seguir apresenta a relação entre alguns desses atributos e classes. Como pode ser observado, esse problema envolve muitos atributos (alguns foram removidos da imagem para facilitar a visualização).

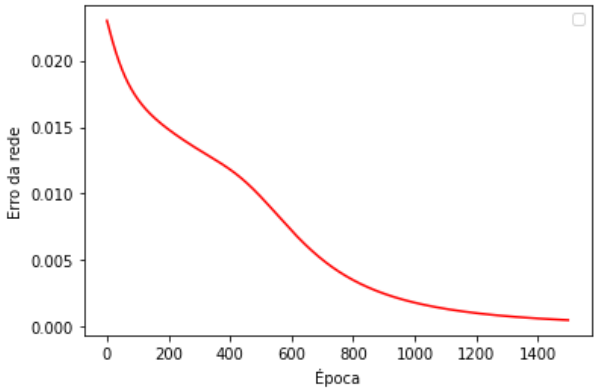


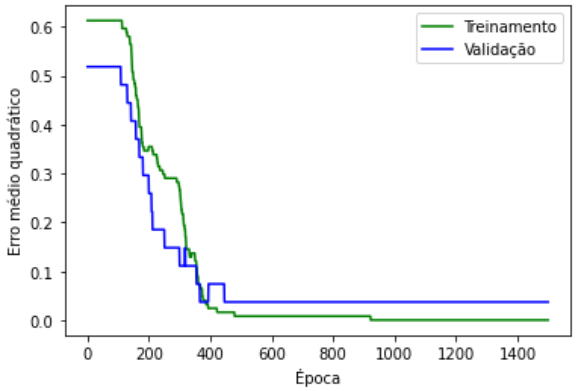
A rede apresenta a seguinte configuração em relação a camadas e neurônios: a camada de entrada apresenta 13 neurônios; a camada oculta, 6 neurônios; a camada de saída, 3 neurônios. Os dados também foram divididos na proporção: 70% treinamento, 15% validação e 15% teste.

A seguir, são apresentados os testes realizados com a rede com diferentes taxas de aprendizado.

Assim como no exemplo anterior (*Íris*) o primeiro teste a seguir apresenta uma execução da rede com a seguinte configuração: taxa de aprendizado 0.001, com 1500 épocas e pesos aleatórios entre -0.1 e 0.1. Os valores obtidos na última iteração foram:

- Erro da rede: 0.000493
- MSE treinamento: 0.0
- MSE validação: 0.037037





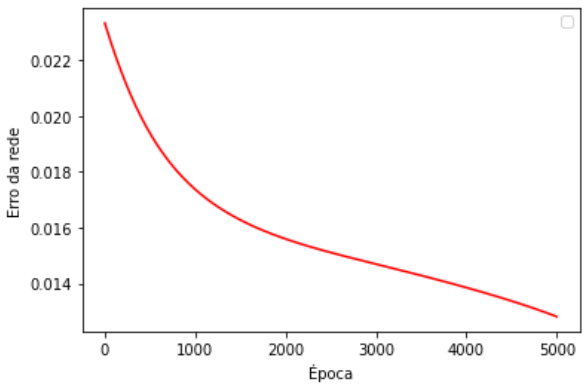
Apresentando novos dados para a rede, ou seja, os dados de teste, a matriz de confusão abaixo é obtida.

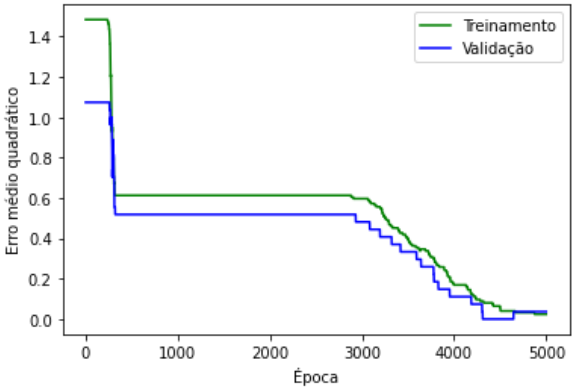
	Desejado		
Obtido	**Classe 1**	**Classe 2**	**Classe 3**
Classe 1	7	0	0
Classe 2	0	10	1
Classe 3	0	0	9

O MSE desse conjunto de teste foi de 0.037037, a taxa de acerto de 96.30% e a de erro, 3.70%.

Reduzindo a taxa de aprendizado para 0.0001 e aumentando a quantidade de épocas para 5000 iterações, observa-se, pelos valores apresentados abaixo, que essa quantidade de iterações não foram ideias para permitir uma convergência/estabilidade da rede:

- Erro da rede: 0.012808
- MSE treinamento: 0.024193
- MSE validação: 0.037037





Observa-se que a partir de 4500 iterações aproximadamente, o MSE do conjunto de treinamento e validação começou a se aproximar.

Utilizando a rede para classificar os dados de teste, foi obtida a seguinte matriz de confusão:

	Desejado		
Obtido	**Classe 1**	**Classe 2**	**Classe 3**
Classe 1	7	2	0
Classe 2	0	8	0
Classe 3	0	0	10

Nesse conjunto de dados de teste o MSE obtido foi de 0.111111. A taxa de acerto foi de 92.60% e a de erro 7.40%. Realizando o mesmo processo com o conjunto de treinamento, temos:

	Desejado		
Obtido	**Classe 1**	**Classe 2**	**Classe 3**
Classe 1	41	1	0
Classe 2	1	47	1
Classe 3	0	0	33

O MSE para esse conjunto de treinamento foi de 0.024193, a taxa de acerto 97.58% e a de erro 2.42%. Pelo fato de ser o conjunto utilizado para o treinamento, esse resultado foi melhor que o anterior. Por fim, realizando a mesma análise para o conjunto de validação, foram obtidos os seguintes valores:

	Desejado		
Obtido	**Classe 1**	**Classe 2**	**Classe 3**
Classe 1	10	0	0
Classe 2	0	12	0
Classe 3	0	1	4

O MSE foi de 0.037037. Já a taxa de erro obtida foi de 3.70% e a de acerto 96.30%.

O próximo exemplo apresenta a execução da mesma configuração diversas vezes, para diferentes taxas de aprendizado. Nesse exemplo, é possível observar a média do erro da rede, a média do MSE de treinamento, validação e teste para cada lote de execução. Para o MSE de treinamento e validação é selecionado aquele obtido na última época. Dessa forma, os valores apresentados representam a média obtida após a execução total da configuração para a determinada taxa de aprendizado.

A rede foi configurada para receber pesos iniciais aleatórios no intervalo entre -0.1 e 0.1. Além disso, foi definido o uso de 900 épocas. Foram testadas três taxas de aprendizado e essa configuração foi executada 30 vezes. Os valores abaixo apresentam a média dos valores após essa execução, para as três taxas.

Taxa de aprendizado: 0.0001

```
 -> Erro da rede (avg): 0.016419
 -> MSE treinamento (avg): 0.580645
 -> MSE validação (avg): 0.592593
```

```
 -> MSE teste (avg): 0.703704
```

Taxa de aprendizado: 0.001

```
 -> Erro da rede (avg): 0.005357
 -> MSE treinamento (avg): 0.002419
 -> MSE validação (avg): 0.025926
 -> MSE teste (avg): 0.038272
```

Taxa de aprendizado: 0.01

```
 -> Erro da rede (avg): 3.4e-05
 -> MSE treinamento (avg): 0.0
 -> MSE validação (avg): 0.037037
 -> MSE teste (avg): 0.019753
```

A situação do mesmo exemplo para o conjunto de dados anterior também acontece para esse. Com o aumento da taxa de aprendizado, a rede tende a convergir com menos iterações. Entretanto, um ponto a ser observado nesse exemplo é que com a taxa de aprendizado 0.01 (que obteve um erro da rede bem pequeno, assim como um MSE de treinamento nulo) o MSE da validação foi pior que aquele obtido com a taxa de aprendizado anterior, podendo ser um indicador de *overfitting*. Para a taxa de aprendizado menor (0.0001), uma quantidade maior de épocas seria necessária para que a convergência acontecesse.

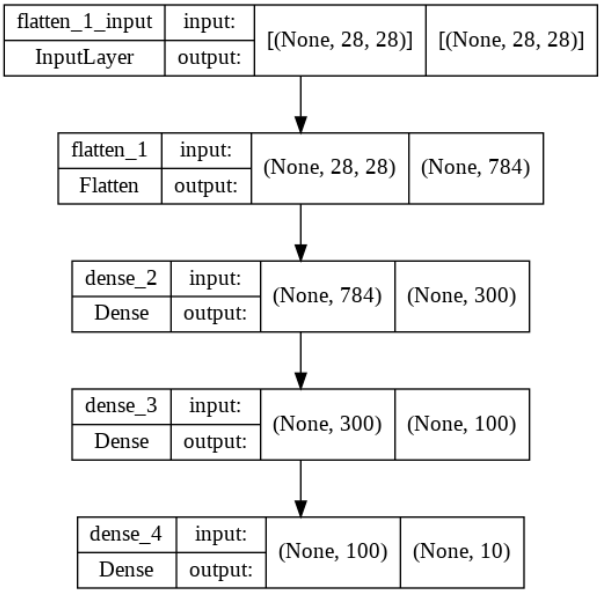
Classificação de imagens

Essa seção apresenta o treinamento de redes neurais para a classificação de imagens. O primeiro exemplo utiliza uma Rede *Multilayer Perceptron* com um conjunto de dados de roupas – *Fashion MNIST*. Já o segundo, uma Rede Neural Convolucional com um conjunto de dados de flores – *Flowers*. Ambos os exemplos utilizam o pacote Keras para a definição e criação dos modelos.

Conjunto de dados *Fashion MNIST*

O conjunto de dados *Fashion MNIST* consiste em 70.000 imagens de 10 tipos de itens de roupa em tons de cinza com dimensões 28 x 28 pixels. As classes desse problema são: `"T-shirt/top"`, `"Trouser"`, `"Pullover"`, `"Dress"`, `"Coat"`, `"Sandal"`, `"Shirt"`, `"Sneaker"`, `"Bag"`, `"Ankle boot"`. A Figura a seguir apresenta amostras dessas classes.

A rede utilizada para esse problema de classificação, apresentada na Figura abaixo possui um modelo chamado *Sequential*. Esse é um tipo de modelo de rede neural disponibilizado pelo pacote do *Keras* e é composto por uma única pilha de camadas conectadas sequencialmente. Em seguida é utilizada uma camada do tipo *Flatten*. Essa camada é responsável por converter cada imagem de entrada em um vetor unidimensional. O método recebe como parâmetro o tamanho da imagem. Ainda, duas camadas ocultas do tipo *Dense* são colocadas no modelo com função de ativação *ReLU* e com 300 e 100 neurônios, respectivamente. Cada camada desse tipo gerencia sua própria matriz de pesos contendo todas as conexões de pesos entre os neurônios e suas entradas. Também existe um gerenciamento do vetor de bias (um por neurônio). Por fim, uma outra camada do tipo *Dense* é colocada como camada de saída. Essa camada tem 10 neurônios justamente por conta da quantidade de classes e a função do tipo *Softmax* porque as classes são exclusivas.

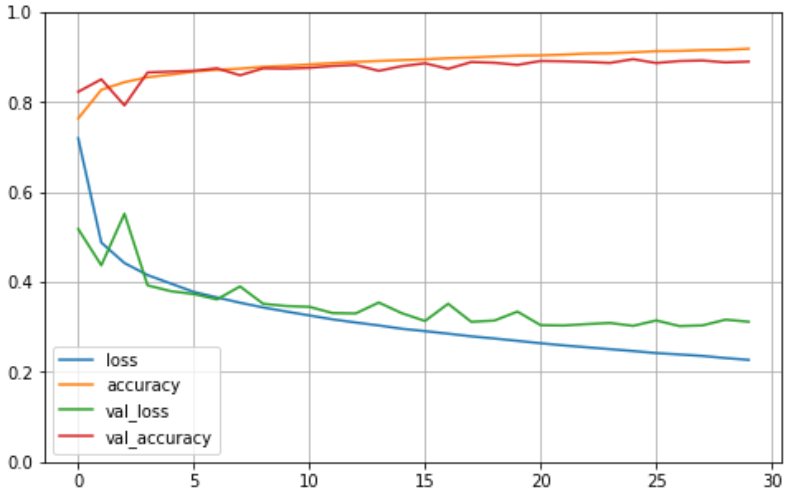


O modelo é compilado com a função de perda/*loss* "`sparse_categorical_crossentropy`" (já que para cada instância do problema existe apenas um índice de classe alvo de 0 a 9, e essas classes são exclusivas), otimizador "`sgd`"

já que o treinamento será feito pelo algoritmo de *Backpropagation*. O treinamento do modelo foi feito com 30 épocas obtendo-se os seguintes valores das métricas na última época:

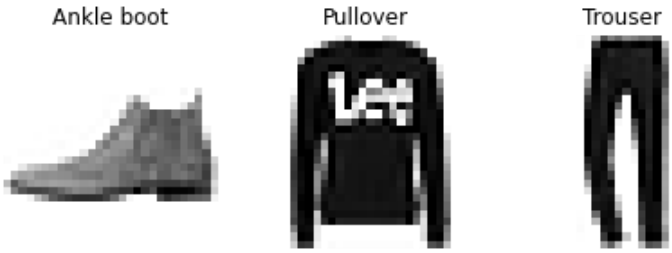
	Treinamento	**Validação**
Loss	0.2270	0.3118
Acurácia	0.9187	0.8902

Se a performance do conjunto de treinamento é melhor que o conjunto de validação, o modelo tem mais chance de gerar *overfitting* no conjunto de treinamento. A acurácia da validação atingiu 89.02% após as 30 épocas e a do treinamento foi de 91.87%, bem próxima da de treinamento, não evidenciando muito *overfitting*. A Figura a seguir mostra os valores das métricas durante o processo do treinamento.



Observa-se que as validações aumentam durante as épocas, enquanto a perda diminui. Comparando as curvas de treinamento e validação, as curvas de validação estão próximas das curvas de treinamento, indicando que não existe muito *overfitting*. Durante o processo de treinamento o erro da validação é calculado no final de cada época, enquanto o erro de treinamento é calculado usando uma média de execução durante cada época. Isso explica as diferenças entre os dois pares de curvas. O desempenho do conjunto de treinamento acaba superando o desempenho da validação, algo que acontece quando o treinamento é feito por tempo suficiente.

Aplicando o conjunto de teste e avaliando o modelo, observa-se que a rede obteve uma perda/*loss* de 33.81% e acurácia de 88.07%. Exibindo a previsão dos três primeiros dados do conjunto de teste, observa-se que a rede acertou as respectivas classes. Para a primeira imagem, obteve-se 96% de chance de pertencer à classe 9 (Ankle boot). Para a segunda, 98% de ser da classe 2 (Pullover). Para a última, 100% de ser Trouser (classe 1). A Figura a seguir mostra o resultado.

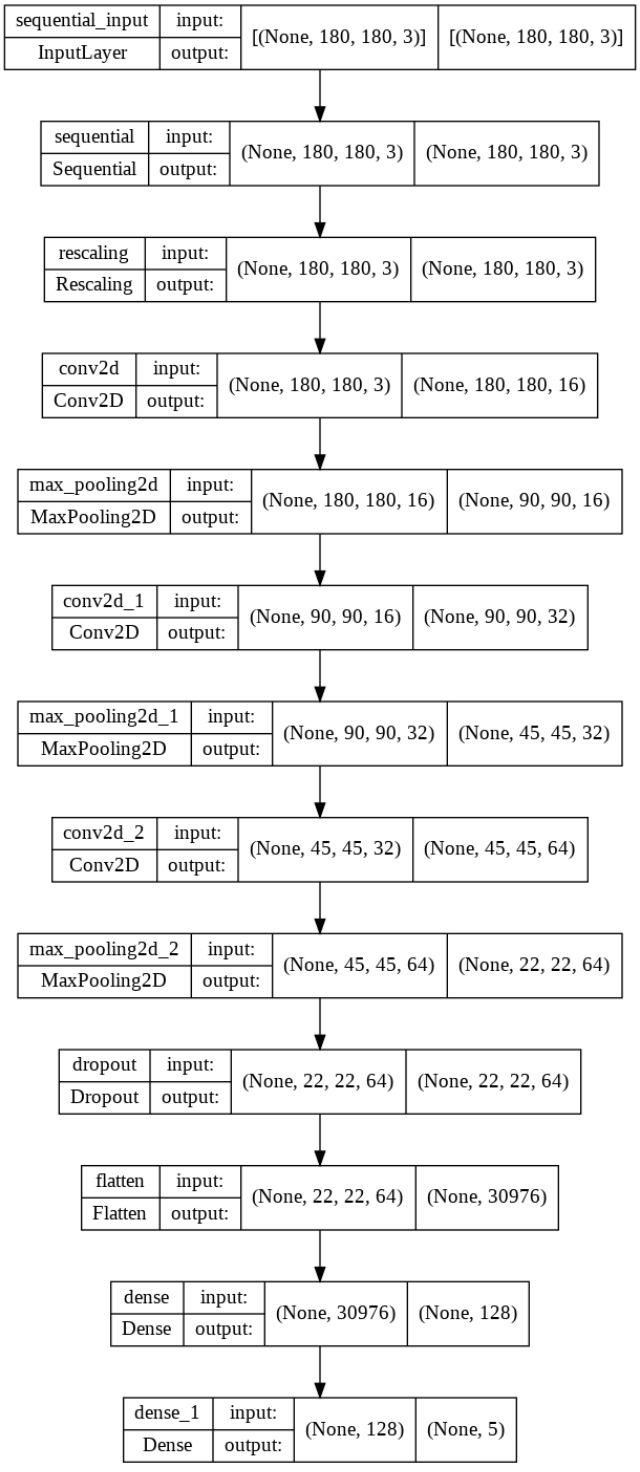


Conjunto de dados de *Flower*

Esse último exemplo, apesar de ter um escopo semelhante ao anterior, treina uma Rede Neural Artificial para classificação de imagens coloridas. O conjunto de dados *Flower* possui 3670 imagens coloridas de flores que pertencem à 5 classes: `'daisy'`, `'dandelion'`, `'roses'`, `'sunflowers'`, `'tulips'`. A Figura a seguir apresenta amostras dessas classes.

O modelo dessa rede é mais complexo que a anterior, devido à complexidade dos dados e do tipo da rede utilizada. A Figura a seguir mostra uma representação gráfica do modelo. Uma camada de aumento de dados é utilizada de modo a evitar *overfitting* (acontece quando existe um número pequeno de dados de treinamento, por exemplo). Essa camada gera dados de treinamento adicionais a partir dos exemplos já existentes. Tais dados são gerados por meio de transformações aleatórias que produzem outras imagens de aparência relevante. Esse processo expõe à rede/modelo a mais aspectos dos dados, aumentando a capacidade de generalização. Assim como no exemplo anterior, as imagens encontram-se no intervalo [0, 255] e devem ser mapeadas para o intervalo [0, 1]. Para isso uma camada do tipo *Rescaling* é utilizada. É acrescentado no modelo *Sequential* três blocos de convolução (*Conv2D*) com uma camada de agrupamento máximo (*MaxPooling2D*) em cada um deles. Há

uma camada totalmente conectada (*Dense*) com 128 unidades em cima dela que é ativada por uma função de ativação *ReLU*. A camada *Dropout* é responsável por buscar reduzir o *overfitting* da rede. Quando é utilizado esse tipo de regularização, a rede descarta aleatoriamente um número de unidades de saída da camada durante o processo de treinamento. O valor passado como parâmetro corresponde à porcentagem de unidades de saídas que serão descartadas aleatoriamente.



Esse modelo também é compilado com a função de perda/*loss* "`sparse_categorical_crossentropy`". O otimizador nesse caso foi o "`adam`", que consiste em uma estimativa de momento adaptável, é outra maneira de usar gradientes passados para calcular gradientes atuais. O treinamento do modelo foi feito com 15 épocas obtendo-se os seguintes valores das métricas na última época:

	Treinamento	**Validação**
Loss	0.4821	0.7818
Acurácia	0.8181	0.7262

Os resultados desse exemplo não foram tão eficientes quanto do anterior. Analisando graficamente, na Figura abaixo, observa-se que, por conta das estratégias descritas anteriormente, o *overfitting* foi menor do que se não tivesse utilizado algumas camadas na rede.

Classificando um exemplo que não foi utilizado durante o treinamento, observa-se que foi definido com 99.81% de exatidão, que a flor da imagem pertence à classe `'sunflowers'`.

Referências

- Aula: Neuro computação – O Sistema Nervoso. Professor Fabricio Breve.
- Aula: Redes Neurais Artificiais (Parte 1 e 2). Professor Fabricio Breve.
- Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition. Disponível em: https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/.
- Classificação de imagem - Tensorflow. Disponível em: https://www.tensorflow.org/tutorials/images/classification.