IESB

Especialização em Inteligência Artificial

Disciplina: Fundamentos de Inteligência Artificial

Docente: Tatiana Tavares

Discente: Henrique Brandão

Proposta:

Utilizando a toolbox de sua preferência, realize o treinamento de ur Perceptron Multicamadas para resolver um problema de classificaçã hiper-parâmetros do seu modelo (número de neurônios, número de aprendizagem, função de ativação, entre outros) e discuta aspectos (porcentagem de classificaçãocorreta).

In [1]:
```
import numpy as np
import pandas as pd

from random import choice

from keras.utils import np_utils
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.backend import clear_session

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
```

Primeiramente, vamos carregar nossos dados para a memória:

In [2]:
```
# Dados podem ser obtidos em: https://archive.ics.uci.edu/ml/dataset
df = pd.read_csv(filepath_or_buffer='iris.csv')
df.shape
```

```
(150, 5)
```

In [3]:
```
df.head()
```

	sepal length	sepal width	petal length	petal width	class
0	5.1	3.5	1.4	0.2	Iris-setosa

	sepal length	sepal width	petal length	petal width	class
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa

Nosso *dataset* tem 5 colunas, quatro (*sepal length*, *sepal width*, *pet* representam características relativas ao nossos registros enquanto categoria (classe) do registro.

Vamos separar os dados de entrada (`features`) e a classe (`target`) treinamento e validação (teste) para começar a construir e testar n

In [4]:
```
x = df.drop(labels=['class'], axis=1)
y = df['class']
x.shape, y.shape
```

```
((150, 4), (150,))
```

Como nosso problema consiste em uma classificação com várias pc transformar nosso dado categórico para um formato vetorial, onde significará o **valor real** da nossa classe:

In [5]:
```
label_encoder = LabelEncoder()
```

In [6]:
```
_ = label_encoder.fit_transform(y)
```

In [7]:
```
y_vec = np_utils.to_categorical(_)
y_vec.shape
```

```
(150, 3)
```

In [8]:
```
label_encoder.classes_
```

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [9]:
```
x.iloc[0], y_vec[0]
```

```
(sepal length    5.1
 sepal width     3.5
 petal length    1.4
 petal width     0.2
 Name: 0, dtype: float64,
 array([1., 0., 0.], dtype=float32))
```

Nossa classe agora está em um formato vetorial, para retorná-la ao que o processo ocorreu de forma fidedigna, podemos fazer o seguir

In [10]:
```
indice = choice(range(x.shape[0]))
_ = df['class'].iloc[indice] == label_encoder.classes_[np.argmax(y_v
print(f'Verificando o índice aleatório #{indice}: {_}')
```

```
Verificando o índice aleatório #129: True
```

In [11]:
```
xtrain, xtest, ytrain, ytest = train_test_split(x, y_vec, test_size=

f'xtrain: {xtrain.shape}, ytrain: {ytrain.shape}, xtest: {xtest.shap
```

```
'xtrain: (112, 4), ytrain: (112, 3), xtest: (38, 4), ytest: (38, 3)'
```

Rede Neural

Vamos criar uma rede neural simples e avaliar seu desempenho no

In [12]:
```
def rede_sequencial(units):
    clear_session()
    clf = Sequential()
    # entrada
    clf.add(Dense(units=units, activation='relu', input_dim=4))
    # oculta #1
    clf.add(Dense(units=units, activation='relu'))
    # oculta #2
    clf.add(Dense(units=units, activation='relu'))
    # saida
    clf.add(Dense(units=3, activation='softmax'))
    clf.compile(optimizer='adam', loss='categorical_crossentropy', m
    print(clf.summary())
    return clf
```

Realizando o treino da rede:

In [13]:
```
nn = rede_sequencial(units=8)
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 8)                 40
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 72
_________________________________________________________________
```

In [14]:
```
nn.fit(x=xtrain, y=ytrain, epochs=1400, shuffle=True)
```

```
Epoch 1/1400
4/4 [==============================] - 7s 1ms/step - loss: 4.5649 - categorical_accurac
Epoch 2/1400
4/4 [==============================] - 0s 1ms/step - loss: 4.1329 - categorical_accurac
Epoch 3/1400
4/4 [==============================] - 0s 1ms/step - loss: 3.7160 - categorical_accurac
Epoch 4/1400
4/4 [==============================] - 0s 2ms/step - loss: 3.6143 - categorical_accurac
Epoch 5/1400
4/4 [==============================] - 0s 1ms/step - loss: 3.2846 - categorical_accurac
Epoch 6/1400
4/4 [==============================] - 0s 1ms/step - loss: 3.1763 - categorical_accurac
Epoch 7/1400
4/4 [==============================] - 0s 830us/step - loss: 2.9594 - categorical_accur
Epoch 8/1400
4/4 [==============================] - 0s 1ms/step - loss: 2.6434 - categorical_accurac
Epoch 9/1400
4/4 [==============================] - 0s 2ms/step - loss: 2.7329 - categorical_accurac
Epoch 10/1400
4/4 [==============================] - 0s 935us/step - loss: 2.4668 - categorical_accur
Epoch 11/1400
4/4 [==============================] - 0s 1ms/step - loss: 2.2244 - categorical_accurac
Epoch 12/1400
4/4 [==============================] - 0s 788us/step - loss: 2.1254 - categorical_accur
```

Testando o desempenho na segunda parte dos dados:

In [15]:
```
pred = nn.predict(x=xtest)
pred[:5]
```

```
array([[8.3312619e-09, 1.6517485e-02, 9.8348248e-01],
       [5.1892589e-06, 7.6877880e-01, 2.3121603e-01],
       [9.9986649e-01, 1.3354323e-04, 1.8363607e-12],
       [5.8906767e-06, 8.8039225e-01, 1.1960184e-01],
       [1.3812244e-08, 6.6447565e-03, 9.9335527e-01]], dtype=float32)
```

Vamos converter os dados de saída da predição para um formato u
classe vetorizada:

In [16]:
```
pred_vec = [np.argmax(x) for x in pred]
pred_vec[:5]
```

```
[2, 1, 0, 1, 2]
```

In [17]:
```
ytest_vec = [np.argmax(x) for x in ytest]
ytest_vec[:5]
```

```
[2, 1, 0, 1, 2]
```

In [18]:
```
print(f'# Matriz de confusão:\n{confusion_matrix(y_true=ytest_vec, y
```

```
# Matriz de confusão:
[[11  0  0]
 [ 0 14  0]
 [ 0  0 13]]
```

In [19]:
```
print(f'# Relatório de classificação:\n{classification_report(y_true=
```

```
# Relatório de classificação:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00        14
           2       1.00      1.00      1.00        13

    accuracy                           1.00        38
   macro avg       1.00      1.00      1.00        38
weighted avg       1.00      1.00      1.00        38
```

Podemos observar que todas as métricas do nosso modelo alcançar

- Contextualizar sua classificação, qual o problema a ser solucion

  > Dadas as medidas relativas às pétalas e sépalas das flores,
  > vel classificá-las através de um algoritmo?
  > R: Sim. Podemos observar o bom desempenho desse modelo.

- Levantar as dificuldades, limitações, inconsistências e imprecis

  > Poucos dados no dataset.

- Sugerir futuras implementações.

  > Testar o modelo em mais registros.

- Para o seu banco de dados existem outros trabalhos de classific
  os.

  > Sim, bastante. Esse dataset é comumente utilizado para inic
  > so de aprendizado de redes neurais.

In [ ]: