Inteligência Artificial - IC/UFF - 2016.1

Trabalho 2. Reconhecendo Flores usando MLPs

Introdução

O segundo trabalho prático tem a ver com um problema clássico dentro da Inteligência Artificial: o reconhecimento de distintas flores de tipo Iris a partir do *Iris flower dataset* https://en.wikipedia.org/wiki/Iris_flower_data_set.

O dataset de flores íris foi originalmente proposto por Ronald Fisher https://en.wikipedia.org/wiki/Ronald_Fisher no artigo de 1936 *"The use of multiple measurements in taxonomic problems as an example of linear discriminant analysis"*.

- O dataset contem 50 amostras de cada espécie de íris: *iris setosa*, *iris virginica* e *iris versicolor*.
- Para cada amostra (flor) foram medidas (em centímetros) quatro características: o comprimento da sépala, a largura da sépala, o comprimento das pétalas e a largura das pétalas.
- Fisher, baseando-se nessas medições, desenvolveu um modelo linear discriminante que com capacidade para distinguir casa espécie das outras.

Descrição do problema

Se queremos projetar um algoritmo para reconhecer as espécies de íris, quais podem ser os dados?

- Precisamos um arranjo de duas dimensões `[n_samples x n_features]`.

Pergunta 1: A que ser refere `n_samples`?

Pergunta 2: Qual é o significado de `n_features`?

Nota: Lembre-se que deve haver um número de propriedades é fixo para cada amostra.

Carregamento dos dados usando `scikit-learn`

Ante tudo precisamos importar o módulo `__future__` para permitir a compatibilidade com Python 2 e

3.

```python
from __future__ import absolute_import, division, print_function, unicode_literals
```

`scikit-learn` permite carregar de uma forma fácil os dados das espécies de íris. `scikit-learn` inclui um arquivo `.csv` e provê a função `load_iris()` que devolve os dados como um conjunto de arrays.

```python
from sklearn.datasets import load_iris
iris_data = load_iris()
```

A variável `iris_data` é uma instancia da classe `Bunch`, que pode ser vista como um `dict` melhorado. Por exemplo, podemos ver as chaves fazendo:

```python
print(iris_data.keys())
```

```
dict_keys(['DESCR', 'data', 'feature_names', 'target_names', 'target'])
```

Também é possível dar um olho na descrição do problema fazendo:

```python
print(iris_data['DESCR'])
```

```
Iris Plants Database
====================

Notes
-----
Data Set Characteristics:
    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica
    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
```

```
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20  0.76     0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

This is a copy of UCI ML iris datasets.
http://archive.ics.uci.edu/ml/datasets/Iris

The famous Iris database, first used by Sir R.A Fisher

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

References
----------
   - Fisher,R.A. "The use of multiple measurements in taxonomic problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda,R.O., & Hart,P.E. (1973) Pattern Classification and Scene Analysis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
     Structure and Classification Rule for Recognition in Partially Exposed
     Environments".  IEEE Transactions on Pattern Analysis and Machine
     Intelligence, Vol. PAMI-2, No. 1, 67-71.
   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
     on Information Theory, May 1972, 431-433.
   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
     conceptual clustering system finds 3 classes in the data.
   - Many, many more ...
```

`iris_data.data` contem os valores das entradas (as X's nas slides da disciplina) e `iris_data.target` contem os valores esperados correspondentes (nossas y's)

Python

```
n_samples, n_features = iris_data.data.shape
print('Número de amostras de entrada:', n_samples)
print('Número de atributo em cada amostra de entrada:', n_features)
print('A primeira amostra:', iris_data.data[0])
```

```
Número de amostras de entrada: 150
Número de atributo em cada amostra de entrada: 4
A primeira amostra: [ 5.1  3.5  1.4  0.2]
```

Python

```
print('Dimensões das entradas:', iris_data.data.shape)
print('Dimensões das classes:', iris_data.target.shape)
```

```
Dimensões das entradas: (150, 4)
Dimensões das classes: (150,)
```

A informação da classe a que corresponde cada amostra de entrada é armazenada em `target` de forma numérica.

Python

```
print(iris_data.target)
```

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

Os nomes correspondentes estão armazenados em `target_names`:

Python

```
print(iris_data.target_names)
```

```
['setosa' 'versicolor' 'virginica']
```

Visualizando o problema

Vamos agora a visualizar a relação entre as propriedades. A biblioteca `seaborn` permite visualizar estas relações muito facilmente.

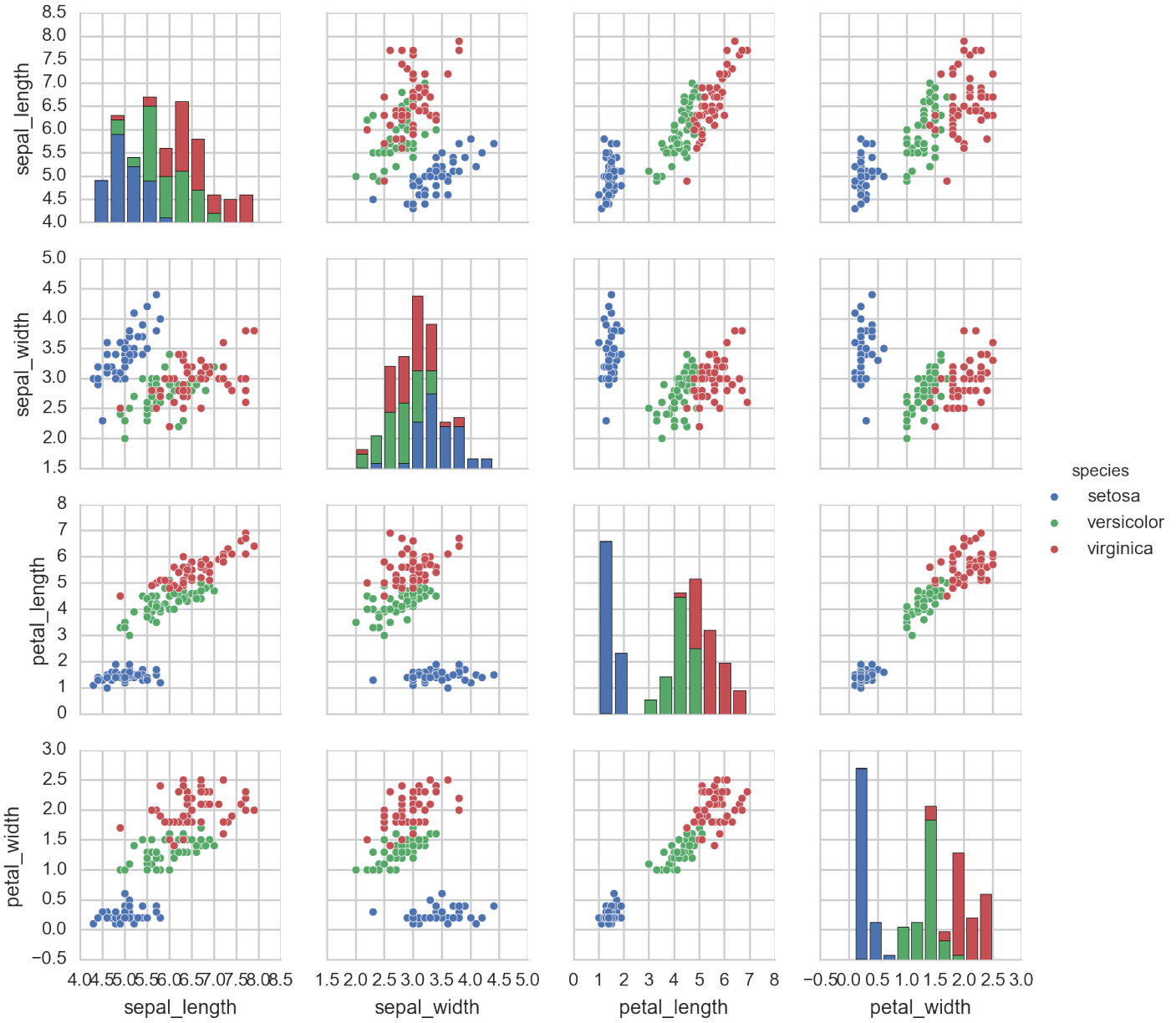
Python

```
# ploting support stuff
import seaborn
seaborn.set_style('whitegrid')

# magic commands for configuring the notebook
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
```

Python

```
seaborn.pairplot(seaborn.load_dataset("iris"), hue="species", size=2);
```



Tudo pronto!

- Vocês tem os dados!

- A tarefa agora é programar um Perceptron Multi-Camada (MLP), seu algoritmo de aprendizagem e ajustar seus hiper-parâmetros para que seja capaz de reconhecer as flores do dataset.

Pergunta 3: Que tipo de problema de aprendizado automático é este?

Pergunta 4: É preciso transformar os dados de alguma forma?

Pergunta 5: Como seria a implementação do ciclo treinamento/validação/teste?

Detalhes da implementação

Você(s) deve(m) implementar uma classe como a seguir:

Python

```
class YourNameMLP():
    def __init__(self, params=None):
        'If params is None then the MLP is initialized with default values.'
        pass

    def propagate(self, X):
        'Propagates the inputs in X returning a list of predictions.'
        pass

    def learn(self, X, y):
        'Performs a learning iteration over the dataset.'
        pass
```

A sua tarefa:

- Responder as perguntas anteriores.
- Entregar a implementação do MLP como um arquivo `.py` seguindo o arquivo de exemplo `trabalho2_seu_nome_mlp.py`.
- Gráficas de erro e progresso serão bem-vindas.
- **Importante:** não pode ser usada na implementação do MLP nenhuma biblioteca pre-existente, como, por exemplo, Keras, theano, TensorFlow, etc.
- Outras bibliotecas (`numpy` , `scikit-learn` , etc.) podem ser usadas como suporte mas não como os algoritmos mesmos.
- O dicionário `params` deve conter a configuração do MLP, número de camadas, número de neurônios por camada, funções de ativação, etc.
- Os valores por defeito de `params` devem corresponder a os melhores valores achados por você(s).
- Os trabalhos podem ser feitos por grupos de 1, 2, ou 3 alunos.

Entregas

- Os trabalhos devem ser entregues via o Google Classroom da disciplina.
- Em caso você tenha mais de um arquivo, entregar como arquivo zip.