Introdução às Redes Neurais Artificiais

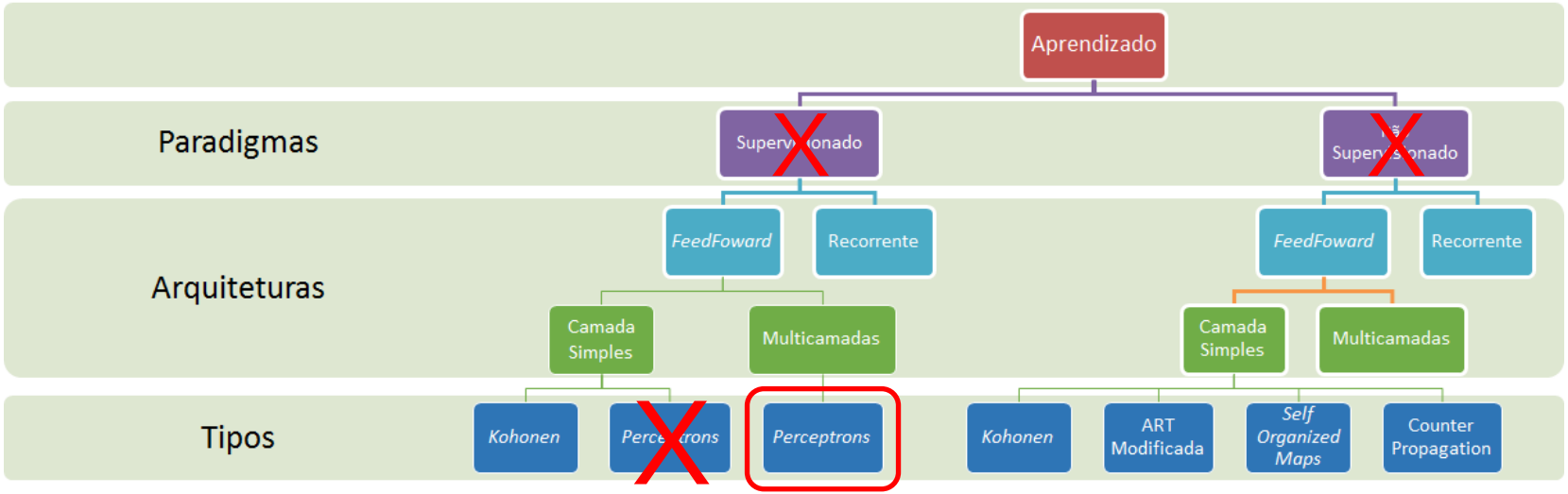
Scikit Learn

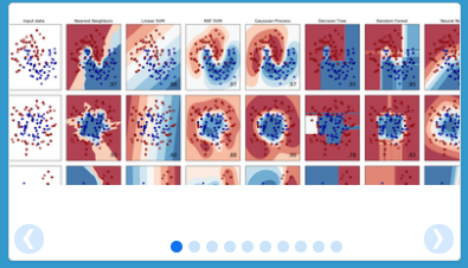
Prof. João Marcos Meirelles da Silva
jmarcos@id.uff.br

Universidade Federal Fluminense

Mapa de Aprendizado

Redução de Dimensionalidade | Tipos de Dados | Detecção de Anomalias





scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Mapa de Aprendizado

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, ...
— Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, ...
— Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, ...
— Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization.
— Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics.
— Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction.
— Examples

News

Community

Who uses scikit-learn?

Classificador MLP

Classe sklearn.neural_network.MLPClassifier

Implementa uma rede Perceptron Multicamadas usando o algoritmo de treinamento *Backpropagation*.

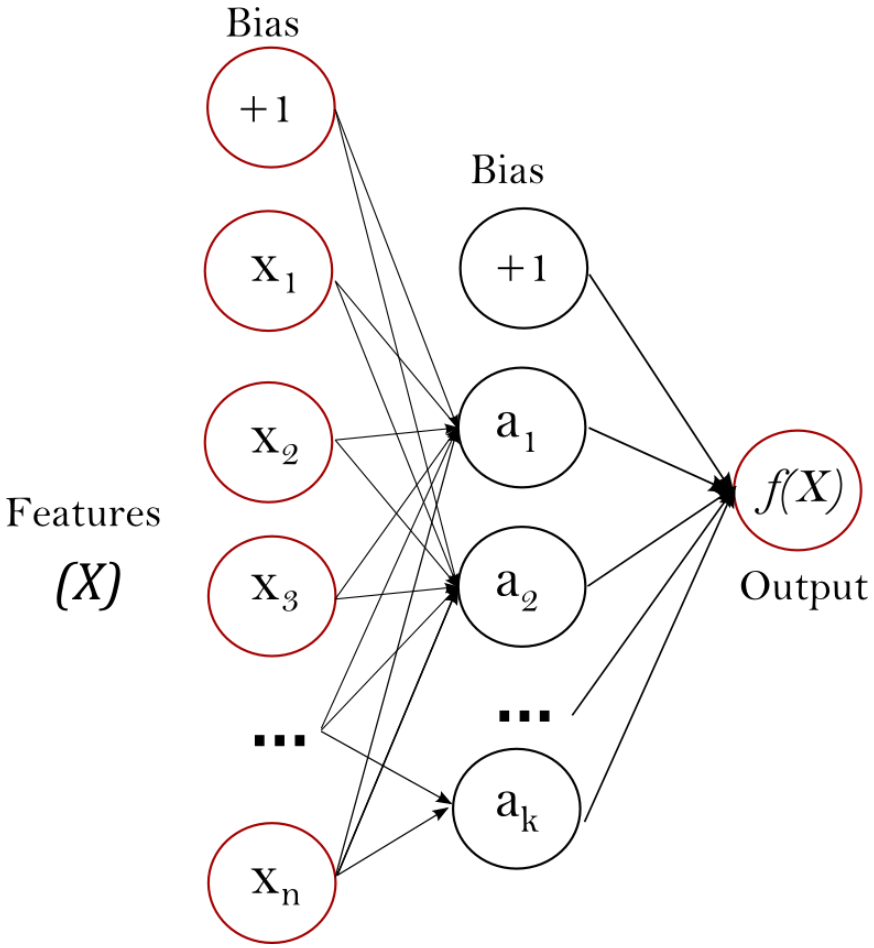
Entrada: Array $X_{(n\_amostras,\ n\_atributos)}$

Saída: Array $Y_{(n\_amostras)}$ → Rótulos de classes

Classificador MLP

Atributos públicos:

- coefs_: Lista de matrizes de pesos, onde a i-ésima matriz de pesos representa os pesos entre a camada i e i+1;
- intercepts_: Lista dos vetores de bias, onde o i-ésimo vetor representa o bias adicionado à camada i+1.



Classificador MLP

```
>>> from sklearn.neural_network import MLPClassifier
>>> X = [[0., 0.], [1., 1.]]
>>> y = [0, 1]
>>> clf = MLPClassifier(solver='lbfgs', alpha=1e-5,
...                     hidden_layer_sizes=(5, 2), random_state=1)
...
>>> clf.fit(X, y)
MLPClassifier(activation='relu', alpha=1e-05, batch_size='auto',
       beta_1=0.9, beta_2=0.999, early_stopping=False,
       epsilon=1e-08, hidden_layer_sizes=(5, 2), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
       solver='lbfgs', tol=0.0001, validation_fraction=0.1, verbose=False,
       warm_start=False)
```

Classificador MLP

Após o comando clf.fit(X,y), a rede está pronta para classificar novas entradas:

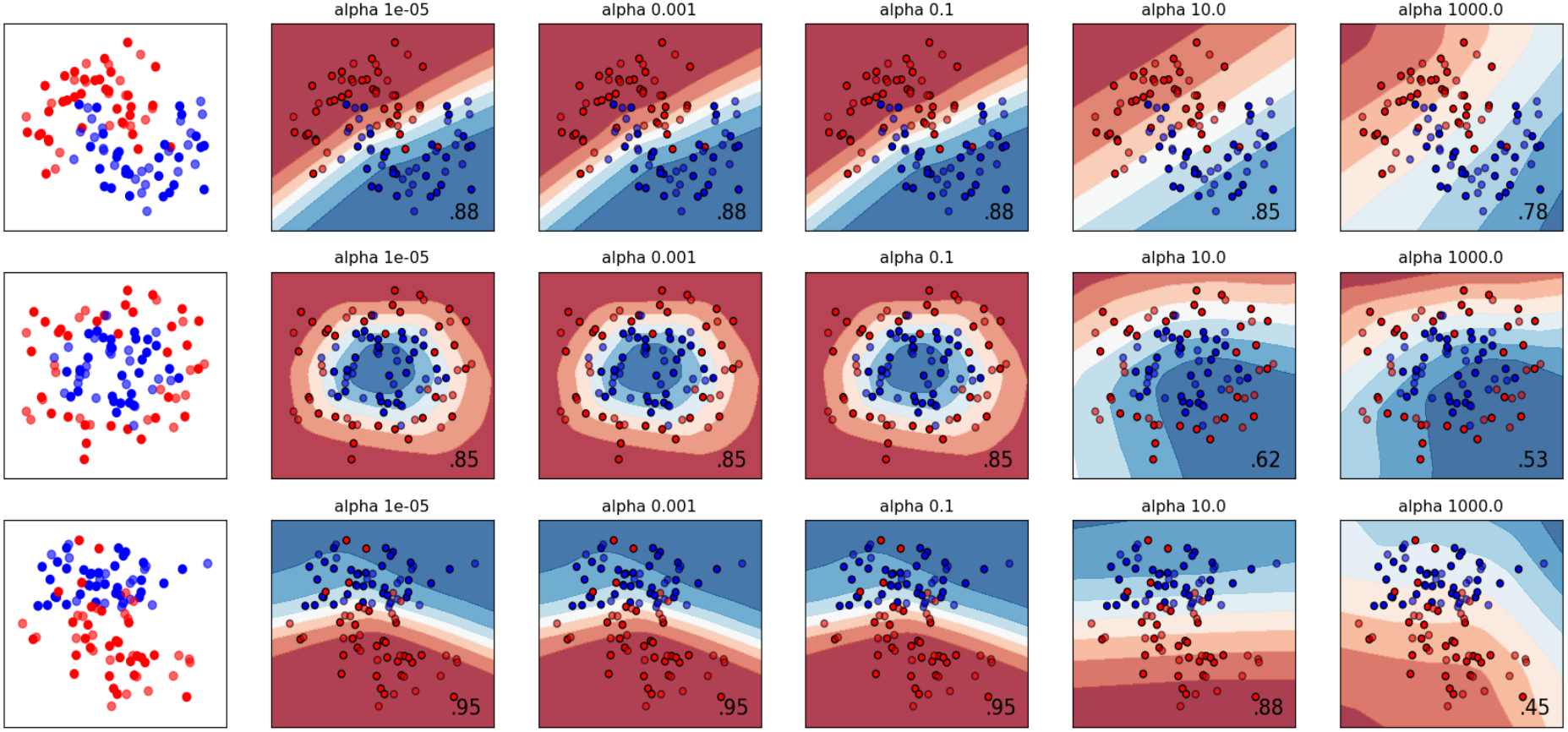
```
>>> clf.predict([[2., 2.], [-1., -2.]])
array([1, 0])
```

O atributo clf.coefs_ contém as matrizes de pesos:

```
>>> [coef.shape for coef in clf.coefs_]
[(2, 5), (5, 2), (2, 1)]
```

Regularização

A classe MLPClassifier usa o parâmetro α para a regularização, o que ajuda a evitar o problema de *overfitting* através da penalização dos pesos com grandes magnitudes.


alpha 1e-05
alpha 0.001
alpha 0.1
alpha 10.0
alpha 1000.0
.88
.88
.88
.85
.78
alpha 1e-05
alpha 0.001
alpha 0.1
alpha 10.0
alpha 1000.0
.85
.85
.85
.62
.53
alpha 1e-05
alpha 0.001
alpha 0.1
alpha 10.0
alpha 1000.0
.95
.95
.95
.88
.45

Algoritmos

A classe MLPClassifier usa os seguintes algoritmos para treinamento:

1. SGD (*Stochastic Gradient Descent*) → online e mini-batch
2. Adam (*Adaptive Moment Estimation*) → online e mini-batch (melhor que SGD)
3. L-BFGS

Dicas

1. MLP é altamente sensível à escala dos atributos. Normalize cada um dos atributos do vetor de entrada para [0,1] ou [-1, +1];

2. Padronize seus dados de modo que a média seja 0 e variância unitária;

3. Use o método *StandardScaler* *(geralmente na escala 10.0**np.arange(1,7))*;
4. Tente encontrar um bom valor para o parâmetro de regularização α (método *GridSearchCV*).

5. Use o método:
   L-BFGS para pequenas bases de dados
   ADAMS para grandes bases de dados

Dicas

```
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler()
>>> # Don't cheat - fit only on training data
>>> scaler.fit(X_train)
>>> X_train = scaler.transform(X_train)
>>> # apply same transformation to test data
>>> X_test = scaler.transform(X_test)
```

PRÁTICA

1) Visitar o site:

   www.scikit-learn.org

2) Clicar em "Classification"

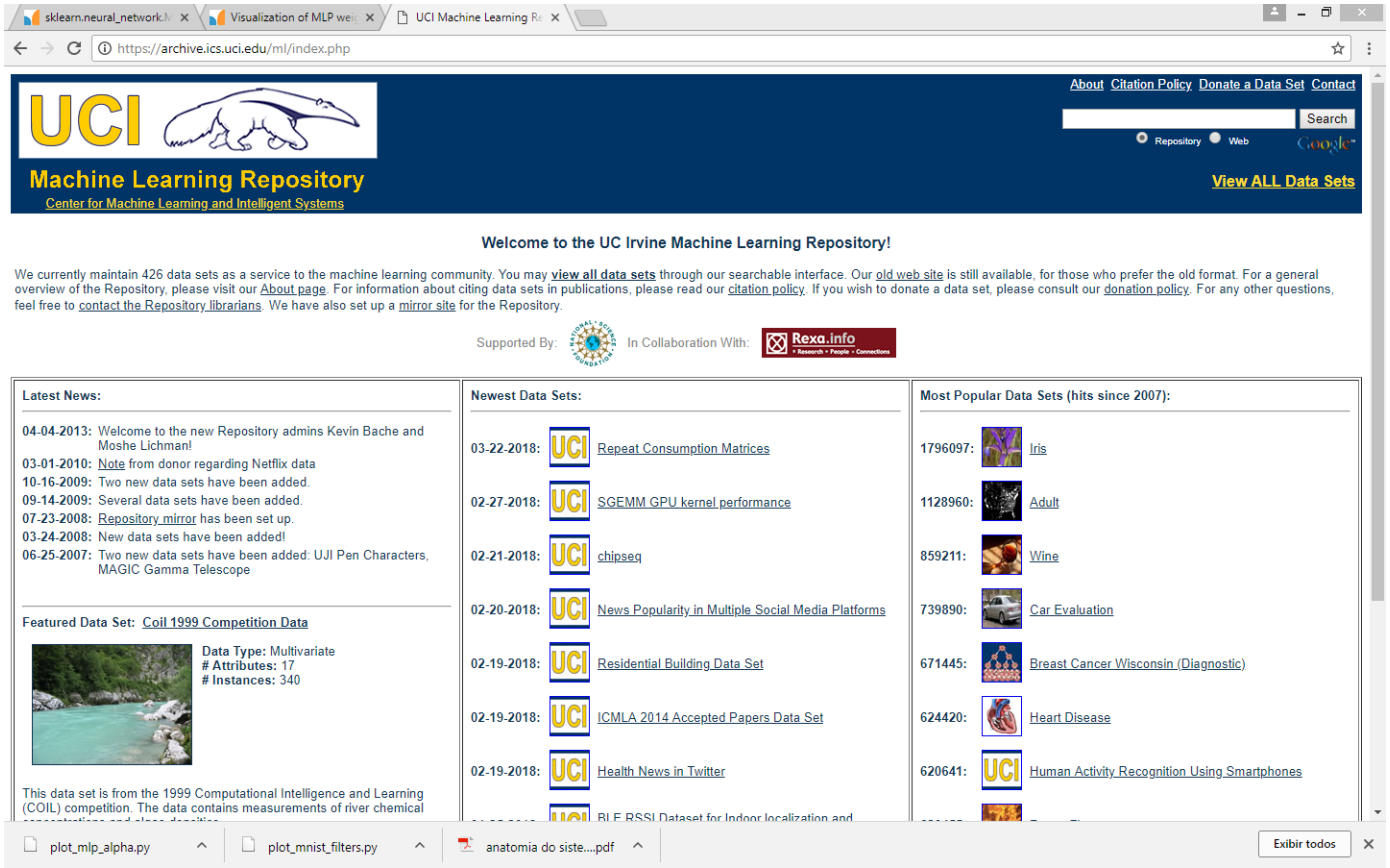
3) Clicar no item "1.17 – Neural Network Models (Supervised)"

4) Leia a página e, por fim, clique em "Visualization of MLP weights on MNIST

5) Leia a página e execute o código exemplo disponível nela.

PRÁTICA

1. Visitar o site
2. Leia sobre a base de dados íris dataset



PRÁTICA

```
# -*- coding: utf-8 -*-
"""
Created on Mon Apr  2 07:49:10 2018

@author: jmarcos
"""

from matplotlib import pyplot as plt
from sklearn.datasets import load_iris
import numpy as np

# We load the data with load_iris from sklearn
data = load_iris()
features = data['data']
feature_names = data['feature_names']
target = data['target']
for t,marker,c in zip(range(3),">ox","rgb"):
    # We plot each class on its own to get different colored markers
    plt.scatter(features[target == t,0],
                features[target == t,1],
                marker=marker, c=c)
```

Crie um classificador usando a classe MLPClassifier para a base da Iris!

Referências

1. *Neural Networks and Learning Machines*, 3rd. Edition, Simon Haykin

2. *Fundamental of Neural Networks - Architectures, Algorithms and Applications*, Laurene Fausett

3. *Pattern Classification*, Richard O. Duda, Peter E. Hart, David G. Stork