A Decision Tree Regressor - Prática

Ciclo	Ciclo 06: Algoritmos baseado em árvores
# Aula	46
Created	@March 18, 2023 8:31 AM
Done	☑
Ready	☑

Objetivo da Aula:

- [ ] Decision Tree Regressor
- [ ] Hiperparametros de controle
- [ ] Resumo
- [ ] Próxima aula

Conteúdo:

▼ 1. Decision Tree Regressor

▼ 1.1 Treinamento do Regressor

```
import cv2
from matplotlib import pyplot as plt
```

```
from sklearn import tree            as tr
from sklearn import datasets        as ds
from sklearn import model_selection as ms
from sklearn import metrics         as mt
from six             import StringIO


# Gerando dados sintéticos
X, y = ds.make_regression(n_features=4, random_state=0)

# Separando dados em treino e teste
X_train, X_test, y_train, y_test = ms.train_test_split(X, y, random_state=0)

# Instanciando e treinando a Decision Tree Regressor
tree_reg = tr.DecisionTreeRegressor(random_state=0)
tree_reg.fit(X_train, y_train)

# Realizando a predição com os dados de teste
y_pred = tree_reg.predict(X_test)

# Calculando o erro médio quadrático (MSE)
mse = mt.mean_squared_error(y_test, y_pred)
print("MSE: ", mse)

# Export draw
dot_data = StringIO()
tr.export_graphviz(
    tree_reg,
    out_file='tree_reg.dot',
    rounded=True,
    filled=True
)

# Convert .dot to .png
!dot -Tpng tree_reg.dot -o tree_reg.png

# Load image on jupyter notebook
img = cv2.imread('tree_reg.png')
plt.figure(figsize = (20, 20))
plt.imshow( img )
```

▼ 1.2 Treinamento do Regressor com Outliers

```
import cv2
from matplotlib import pyplot as plt

from sklearn import tree            as tr
from sklearn import datasets        as ds
from sklearn import model_selection as ms
from sklearn import metrics         as mt
```

```
from six             import StringIO


# Gerando dados sintéticos com ruído
X, y = ds.make_regression(n_features=4, random_state=0, noise=5)

# Separando dados em treino e teste
X_train, X_test, y_train, y_test = ms.train_test_split(X, y, random_state=0)

# Instanciando e treinando a Decision Tree Regressor
tree_reg = tr.DecisionTreeRegressor(random_state=0)
tree_reg.fit(X_train, y_train)

# Realizando a predição com os dados de teste
y_pred = tree_reg.predict(X_test)

# Calculando o erro médio quadrático (MSE)
mse = mt.mean_squared_error(y_test, y_pred)
print("MSE: ", mse)

# Export draw
dot_data = StringIO()
tr.export_graphviz(
    tree_reg,
    out_file='tree_reg.dot',
    rounded=True,
    filled=True
)

# Convert .dot to .png
!dot -Tpng tree_reg.dot -o tree_reg.png

# Load image on jupyter notebook
img = cv2.imread('tree_reg.png')
plt.figure(figsize = (20, 20))
plt.imshow( img )
```

▼ 1.3 Treinamento do Regressor com escalas diferentes

```
import cv2
from matplotlib import pyplot as plt

from sklearn import tree            as tr
from sklearn import datasets        as ds
from sklearn import model_selection as ms
from sklearn import metrics         as mt
from six             import StringIO
```

```
# Gerando dados sintéticos com ruído
X, y = ds.make_regression(n_features=4, random_state=0, noise=5)


# Generate synthetic data with different scales
X1 = np.random.normal(0, 1, size=(100, 1))
X2 = np.random.uniform(-5, 5, size=(100, 1))
X3 = np.random.normal(10, 2, size=(100, 1))
X4 = np.random.uniform(-10, 10, size=(100, 1))
X = np.hstack((X1, X2, X3, X4))


# Separando dados em treino e teste
X_train, X_test, y_train, y_test = ms.train_test_split(X, y, random_state=0)

# Instanciando e treinando a Decision Tree Regressor
tree_reg = tr.DecisionTreeRegressor(random_state=0)
tree_reg.fit(X_train, y_train)

# Realizando a predição com os dados de teste
y_pred = tree_reg.predict(X_test)

# Calculando o erro médio quadrático (MSE)
mse = mt.mean_squared_error(y_test, y_pred)
print("MSE: ", mse)

# Export draw
dot_data = StringIO()
tr.export_graphviz(
    tree_reg,
    out_file='tree_reg.dot',
    rounded=True,
    filled=True
)

# Convert .dot to .png
!dot -Tpng tree_reg.dot -o tree_reg.png

# Load image on jupyter notebook
img = cv2.imread('tree_reg.png')
plt.figure(figsize = (20, 20))
plt.imshow( img )
```

▼ 2. Hiperparametros de controle

A algoritmo Decision Tree é um modelo não-paramétrico. Apesar de possuir parâmetros, seu modelo não depende de uma fórmula pré-estabelecida como as Regressões por exemplo.

A falta de um modelo matemático prévio, fornece um alto grau de liberdade que aumentam as chances de overfitting.

Para reduzir as chances de overfitting é necessário regular alguns parâmetros do algoritmo que controlam o crescimento da árvore ou limitam o número de recortes espaciais feito pelo algoritmo, no conjunto de dados.

Os parâmetros que regulam a Decision Tree são:

- *max_depth:* controle o tamanho máximo das quebras, ou seja, o tamanho máximo do crescimento da árvore ou ainda o número de recortes espaciais.
- *min_samples_leaf:* O número mínimo de amostras do nó deve ter, antes de fazer uma nova separação e gerar nós filhos.
- *min_weight_fraction_leaf:* A mesma definição do parâmetro min_samples_leaf, mas definida como a fração do número total dos pesos das evidências.
- *max_features:* O número máximo de atributos que são avaliados para a divisão de cada nó.

Em regras gerais, aumentando os parâmetros que começam com *min_* ou reduzindo os parâmetros que começam com *max_* vão regular o algoritmo.

▼ 3. Resumo

1. O aprendizado não-supervisionado tem o objetivo de agrupar indivíduos com características ou comportamentos semelhantes, para encontrar padrões.

▼ 4. Próxima aula

Random Forest - Teoria