Data Analysis and Machine Learning: Nearest Neighbors and Decision Trees

Morten Hjorth-Jensen[1,2]

Department of Physics, University of Oslo[1]

Department of Physics and Astronomy and National Superconducting Cyclotron Laboratory, Michigan State University[2]

Nov 1, 2018

Decision trees, overarching aims

Decision trees are supervised learning algorithms used for both, classification and regression tasks where we will concentrate on classification in this first part of our decision tree tutorial. Decision trees are assigned to the information based learning algorithms which use different measures of information gain for learning. We can use decision trees for issues where we have continuous but also categorical input and target features.

Nearest Neighbors

```
import mglearn
import numpy as np
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier

# Generate sample data
X = np.sort(5*np.random.rand(40,1), axis=0)
y = X**3
y=y.ravel()

# Add noise to targets
X[::4] +=(0.5 - np.random.rand(1))
y[::5] +=(0.5 - np.random.rand(8))

a=np.array(X)
b=np.array(y)

X_train=a[:19]
X_test=a[19:]
y_train=b[:19]
y_test=b[19:]

model=Pipeline([('poly', PolynomialFeatures(degree=3)),('linear', Line
model=model.fit(X_train, y_train)
```

Decision trees and Regression

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

steps=250

distance=0
x=0
distance_list=[]
steps_list=[]
while x<steps:
    distance+=np.random.randint(-1,2)
    distance_list.append(distance)
    x+=1
    steps_list.append(x)
plt.plot(steps_list,distance_list, color='green', label="Random Walk D

steps_list=np.asarray(steps_list)
distance_list=np.asarray(distance_list)

X=steps_list[:,np.newaxis]

#Polynomial fits

#Degree 2
poly_features=PolynomialFeatures(degree=2, include_bias=False)
X_poly=poly_features.fit_transform(X)
```