Polynomial Regression

# example code

|  |
| --- |
| # Import list  import numpy as np  import pandas as pd  import datetime as dt  import pylab as pl  import matplotlib.pyplot as plt  from sklearn.linear\_model import LinearRegression  from sklearn.metrics import mean\_squared\_error  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import r2\_score  # Function define  def f(size):      x = np.linspace(0,4.5,size)      y = 2 \* np.sin(x \* 1.5)      # Make sin graph      return (x,y)  def sample(size):      x = np.linspace(0,4.5,size)      y = 2 \* np.sin(x \* 1.5) + pl.randn(x.size)      # Sample data generation based on sin graph.      return (x,y)  def fit\_polynomial(x,y,degree):      model = LinearRegression()      model.fit(np.vander(x,degree+1),y)      # Modeling input data with degree.      return model  def apply\_polynomial(model,x):      degree = model.coef\_.size -1      y = model.predict(np.vander(x,degree + 1))      # Generating a prediction graph based on the model.      return y  def plot\_learning\_curves(model, X, y, X\_train, X\_val, y\_train, y\_val,degree):      train\_errors, val\_errors = [], []      for m in range (1, len(X\_train)):          model.fit(np.vander(X\_train[:m],degree+1), y\_train[:m])          y\_train\_predict = model.predict(np.vander(X\_train[:m],degree+1))          y\_val\_predict = model.predict(np.vander(X\_val,degree+1))          train\_errors.append(mean\_squared\_error(y\_train[:m], y\_train\_predict))          val\_errors.append(mean\_squared\_error(y\_val, y\_val\_predict))      plt.plot(np.sqrt(train\_errors), "r-+", linewidth = 2, label = "train set")      plt.plot(np.sqrt(val\_errors), "b-", linewidth = 3, label = "validation set")      plt.xlabel("size of train set")      plt.ylabel("RMSE (degree {})".format(degree + 1))      plt.legend()      #Draw Learning curves.  # Make sample Data.  f\_x,f\_y = f(1000)  s\_x,s\_y = sample(1000)  # Split sameple Data.  X\_train, X\_val, y\_train, y\_val = train\_test\_split(s\_x,s\_y, test\_size=0.2,random\_state= 0)  # Draw sample Data.  plt.subplot(2,3,1)  plt.plot(s\_x,s\_y,'k.')  # Make model and prediction value(degree 1)  model = fit\_polynomial(X\_train,y\_train,1)  p\_y = apply\_polynomial(model,X\_val)  # Draw prediction graph (degree 1)  plt.subplot(2,3,2)  plt.plot(X\_val,p\_y,'g')  # Make model and prediction value(degree 3)  model\_3 = fit\_polynomial(X\_train,y\_train,3)  p\_y\_3 = apply\_polynomial(model\_3,X\_val)  # Draw prediction graph (degree 3)  plt.subplot(2,3,3)  plt.plot(X\_val,p\_y\_3,'b')  # Draw learning\_curves (degree 1)  plt.subplot(2,3,4)  model\_curve = LinearRegression()  plot\_learning\_curves(model\_curve,s\_x,s\_y,X\_train,X\_val,y\_train,y\_val,1)  # Check r2\_score of prediction value (degree 1)  score\_1 = r2\_score(y\_val,p\_y)  print("r2\_score of sample model (degree 1) : {}".format(score\_1))  # Draw learning\_curves (degree 3)  plt.subplot(2,3,5)  model\_curve = LinearRegression()  plot\_learning\_curves(model\_curve,s\_x,s\_y,X\_train,X\_val,y\_train,y\_val,3)  # Check r2\_score of prediction value (degree 3)  score\_2 = r2\_score(y\_val,p\_y\_3)  print("r2\_score of sample model (degree 3) : {}".format(score\_2)) |

# testing result

|  |
| --- |
|  |