```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Example 2.3
Learning curves
Machine Learning for Engineering Problem Solving
@author: Austin Downey
"""

import IPython as IP
IP.get_ipython().run_line_magic('reset', '-sf')

import numpy as np
import matplotlib.pyplot as plt
import sklearn as sk
from sklearn import linear_model
from sklearn import pipeline

plt.close('all')

#%% build the data sets
np.random.seed(2) # 2 and 6 are pretty good
m = 100
X = 6 * np.random.rand(m,1) - 3
Y = 0.5 * X**2 + X + 2 + np.random.randn(m,1)
X_model = np.linspace(-3,3)

# plot the data
plt.figure()
plt.grid(True)
plt.scatter(X,Y)
plt.xlabel('x')
plt.ylabel('y')

#%% generate learing curves for a linear model

# build the linear model in SK learn
model = sk.linear_model.LinearRegression()

# split the data into training and validation data sets
# Split arrays or matrices into random train and test subsets
X_train, X_val, y_train, y_val = sk.model_selection.train_test_split(X, Y, test_size=0.2)

train_errors, val_errors = [], []
for i in range(1, len(X_train)):
    model.fit(X_train[:i], y_train[:i])
    y_train_predict = model.predict(X_train[:i])
    y_val_predict = model.predict(X_val)

    # compute the error for the trained model
    mse_train = sk.metrics.mean_squared_error(y_train[:i],y_train_predict)
    train_errors.append(mse_train)

    # compute the error for the validation model
    mse_val = sk.metrics.mean_squared_error(y_val,y_val_predict)
    val_errors.append(mse_val)

    # predict model
    y_model = model.predict(np.expand_dims(X_model,axis=1))

    plt.figure('test model')
    plt.scatter(X,Y,s=2, label='data')
    plt.scatter(X_train[:i],y_train[:i], label='data in training set')
    plt.scatter(X_val,y_val, marker='s', label='validation data')
    plt.plot(X_model,y_model,'r--',label='model')
    plt.xlabel('x')
    plt.ylabel('y')
```

```python
    plt.legend(loc=2)
    plt.grid(True)
    plt.savefig('test_plots/linear_model_'+str(i))
    plt.close('test model')

plt.figure()
plt.grid(True)
plt.plot(train_errors, "--",label="train")
plt.plot(val_errors, ":", label="val")
plt.xlabel('number of data points in training set')
plt.ylabel('mean squared error')
plt.legend(framealpha=1)
plt.ylim(0,6)
#%% generate learning curves for a polynomial model

model = sk.pipeline.Pipeline((
("poly_features", sk.preprocessing.PolynomialFeatures(degree=20, include_bias=False)),
("lin_reg", sk.linear_model.LinearRegression()),
))

# split the data into training and validation data sets
# Split arrays or matrices into random train and test subsets
X_train, X_val, y_train, y_val = sk.model_selection.train_test_split(X, Y, test_size=0.2)

train_errors = []
val_errors = []
for i in range(1, len(X_train)):
    model.fit(X_train[:i], y_train[:i])
    y_train_predict = model.predict(X_train[:i])
    y_val_predict = model.predict(X_val)

    # compute the error for the trained model
    mse_train = sk.metrics.mean_squared_error(y_train[:i],y_train_predict)
    train_errors.append(mse_train)

    # compute the error for the validation model
    mse_val = sk.metrics.mean_squared_error(y_val,y_val_predict)
    val_errors.append(mse_val)

    plt.figure('test model')
    plt.scatter(X,Y,s=2, label='data')
    plt.scatter(X_train[:i],y_train[:i], label='data in training set')
    plt.scatter(X_val,y_val, marker='s', label='validation data')
    y_model = model.predict(np.expand_dims(X_model,axis=1))
    plt.plot(X_model,y_model,'r--',label='model')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.legend(loc=2)
    plt.grid(True)
    plt.savefig('test_plots/polynominal_model_'+str(i))
    plt.close('test model')

plt.figure()
plt.grid(True)
plt.plot(train_errors, "--",label="train")
plt.plot(val_errors, ":", label="val")
plt.xlabel('number of data points in training set')
plt.ylabel('mean squared error')
plt.legend(framealpha=1)
plt.ylim(0,6)
```