

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """
4  Example 2.3
5  Learning curves
6  Machine Learning for Engineering Problem Solving
7  @author: Austin Downey
8  """
9
10 import IPython as IP
11 IP.get_ipython().run_line_magic('reset', '-sf')
12
13 import numpy as np
14 import matplotlib.pyplot as plt
15 import sklearn as sk
16 from sklearn import linear_model
17 from sklearn import pipeline
18
19 plt.close('all')
20
21 %% build the data sets
22 np.random.seed(2) # 2 and 6 are pretty good
23 m = 100
24 X = 6 * np.random.rand(m,1) - 3
25 Y = 0.5 * X**2 + X + 2 + np.random.randn(m,1)
26 X_model = np.linspace(-3,3)
27
28 # plot the data
29 plt.figure()
30 plt.grid(True)
31 plt.scatter(X,Y)
32 plt.xlabel('x')
33 plt.ylabel('y')
34
35 %% generate learning curves for a linear model
36
37 # build the linear model in SK learn
38 model = sk.linear_model.LinearRegression()
39
40 # split the data into training and validation data sets
41 # Split arrays or matrices into random train and test subsets
42 X_train, X_val, y_train, y_val = sk.model_selection.train_test_split(X, Y, test_size=0.2)
43
44 train_errors, val_errors = [], []
45 for i in range(1, len(X_train)):
46     model.fit(X_train[:i], y_train[:i])
47     y_train_predict = model.predict(X_train[:i])
48     y_val_predict = model.predict(X_val)
49
50     # compute the error for the trained model
51     mse_train = sk.metrics.mean_squared_error(y_train[:i], y_train_predict)
52     train_errors.append(mse_train)
53
54     # compute the error for the validation model
55     mse_val = sk.metrics.mean_squared_error(y_val, y_val_predict)
56     val_errors.append(mse_val)
57
58 # predict model
59 y_model = model.predict(np.expand_dims(X_model,axis=1))
60
61 plt.figure('test model')
62 plt.scatter(X,Y,s=2, label='data')
63 plt.scatter(X_train[:i],y_train[:i], label='data in training set')
64 plt.scatter(X_val,y_val, marker='s', label='validation data')
65 plt.plot(X_model,y_model,'r--',label='model')
66 plt.xlabel('x')
67 plt.ylabel('y')

```

```

68     plt.legend(loc=2)
69     plt.grid(True)
70     plt.savefig('test_plots/linear_model_'+str(i))
71     plt.close('test model')
72
73 plt.figure()
74 plt.grid(True)
75 plt.plot(train_errors, "--", label="train")
76 plt.plot(val_errors, ":", label="val")
77 plt.xlabel('number of data points in training set')
78 plt.ylabel('mean squared error')
79 plt.legend(framealpha=1)
80 plt.ylim(0,6)
81 ## generate learning curves for a polynomial model
82
83 model = sk.pipeline.Pipeline((
84     ("poly_features", sk.preprocessing.PolynomialFeatures(degree=20, include_bias=False)),
85     ("lin_reg", sk.linear_model.LinearRegression()),
86 ))
87
88 # split the data into training and validation data sets
89 # Split arrays or matrices into random train and test subsets
90 X_train, X_val, y_train, y_val = sk.model_selection.train_test_split(X, Y, test_size=0.2)
91
92 train_errors = []
93 val_errors = []
94 for i in range(1, len(X_train)):
95     model.fit(X_train[:i], y_train[:i])
96     y_train_predict = model.predict(X_train[:i])
97     y_val_predict = model.predict(X_val)
98
99     # compute the error for the trained model
100    mse_train = sk.metrics.mean_squared_error(y_train[:i], y_train_predict)
101    train_errors.append(mse_train)
102
103    # compute the error for the validation model
104    mse_val = sk.metrics.mean_squared_error(y_val, y_val_predict)
105    val_errors.append(mse_val)
106
107    plt.figure('test model')
108    plt.scatter(X, Y, s=2, label='data')
109    plt.scatter(X_train[:i], y_train[:i], label='data in training set')
110    plt.scatter(X_val, y_val, marker='s', label='validation data')
111    y_model = model.predict(np.expand_dims(X_val, axis=1))
112    plt.plot(X_val, y_model, 'r--', label='model')
113    plt.xlabel('x')
114    plt.ylabel('y')
115    plt.legend(loc=2)
116    plt.grid(True)
117    plt.savefig('test_plots/polynominal_model_'+str(i))
118    plt.close('test model')
119
120 plt.figure()
121 plt.grid(True)
122 plt.plot(train_errors, "--", label="train")
123 plt.plot(val_errors, ":", label="val")
124 plt.xlabel('number of data points in training set')
125 plt.ylabel('mean squared error')
126 plt.legend(framealpha=1)
127 plt.ylim(0,6)
128
129
130
131

```