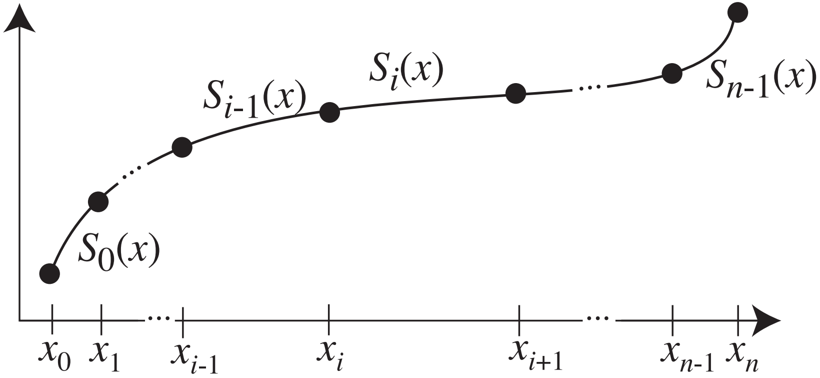
7. Spline Regression

Bruce E. Shapiro

Getting Started in Machine Learning

Last revised: February 14, 2019

Concept of Spline Interpolation



Concept of Spline Interpolation

Given $(x_0, y_0), \dots, (x_{n-1}, y_{n-1})$, where $x_0 < x_1 < \cdots$, fit

$$f(x) = \begin{cases} S_0(x) = b_{00} + b_{01}x + b_{02}x^2 + b_{03}x^3, & x_0 \le x < x_1 \\ S_1(x) = b_{10} + b_{11}x + b_{12}x^2 + b_{13}x^3, & x_1 \le x < x_2 \\ S_2(x) = b_{20} + b_{21}x + b_{22}x^2 + b_{23}x^3, & x_2 \le x < x_3 \\ \vdots & \\ S_{n-2}(x) = b_{n-2,0} + b_{n-2,1}x + b_{n-2,2}x^2 + b_{n-2,3}x^3, & x_{n-2} \le x \le x_{n-1} \end{cases}$$

1. $f(x_i) = y_i$ for $i = 0, \dots, n-1$;
2. $f(x), f'(x), f''(x)$ is continuous at each x_i (**knot points**)
3. $f''(a) = f''(b) = 0$ where $a = x_0$ and $b = x_{n-1}$.

Spline Regression vs Spline Interpolation

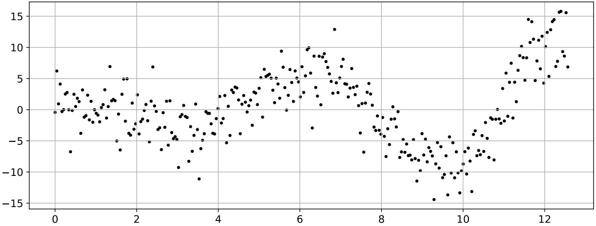
- Let K = number of knots; n = number of points
- $K = n$: interpolation (curve passes through every point)
  - Both the x and y values of the knot points are specified
- $K < n$: regression (typically $K \ll n$)
  - Only the x values, or the number of knots, is specified
  - The method determines the y values, by minimizing an objective function, such as (**smoothed spline**)

$$\mathcal{E} = \lambda \sum_{i=0}^{n-2} \left[\frac{y_i - S_i(x)}{\sigma_i} \right]^2 + (1 - \lambda) \int_a^b f''(x)^2 dx$$

  - With suitable modification, the objective function can fit to fewer S_i than points, and the knot points do not have to match the data points (**free regression spline**)

- **Example (1).** Generate random data (300 points) about $y = x\cos x + N(0,3)$ on $[0, 4\pi]$

```
import numpy as np
import math
X=np.linspace(0,4*math.pi,300)
np.random.seed(99)
n=len(X)
Y=X*np.cos(X)+ np.random.normal(0,3,n)
```



- **Example (2)**. Sorting a list of pairs in Python

```
from sklearn.model_selection import train_test_split

def shakeupdata(X,Y):
        XTRAIN, XTEST, YTRAIN, YTEST=train_test_split(X,Y)
#
# Order the (x,y) pairs in increasing order by x
#
        XYTRAIN=list(zip(XTRAIN,YTRAIN)) # create pairs
        XYTRAIN.sort() # sorts in place
        XTRAIN,YTRAIN=zip(*XYTRAIN) #  unzips using "splat"

        XYTEST=list(zip(XTEST,YTEST))
        XYTEST.sort()
        XTEST,YTEST=zip(*XYTEST) # "splat" operator

    return(XTRAIN,YTRAIN,XTEST,YTEST)
```

- **Example(3).** Generate spline with knots at $x = 2, 6, 8$

```
from scipy.interpolate import LSQUnivariateSpline as LS
from sklearn.metrics import mean_squared_error
fit1=LS(XTRAIN,YTRAIN,[2,6,8])
pred1=fit1(XTEST) # list of predicted y-values
MSE1=mean_squared_error(pred1,YTEST)
print("Mean Squared Error MSE1 =", MSE1)
```

```
Mean Squared Error MSE1 = 11.614392145840794
```

- **Example(3).** Generate spline with knots at $x = 2, 6, 8$

```
from scipy.interpolate import LSQUnivariateSpline as LS
from sklearn.metrics import mean_squared_error
fit1=LS(XTRAIN,YTRAIN,[2,6,8])
pred1=fit1(XTEST) # list of predicted y-values
MSE1=mean_squared_error(pred1,YTEST)
print("Mean Squared Error MSE1 =", MSE1)
```

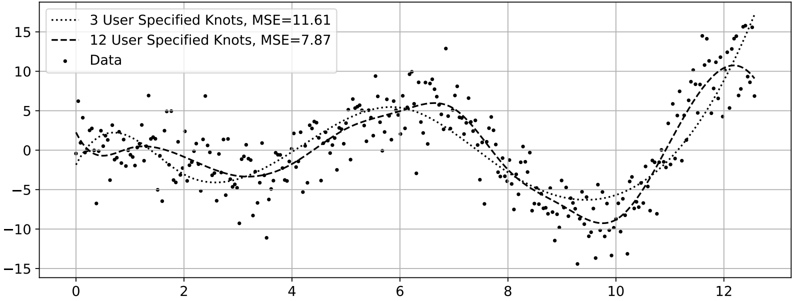
```
Mean Squared Error MSE1 = 11.614392145840794
```

- Generate spline with knots at $x = 1, 2, 3, \ldots, 12$

```
fit2=LS(XTRAIN,YTRAIN,np.arange(1,12,1))
pred2=fit2(XTEST)
MSE2=mean_squared_error(pred2,YTEST)
print("Mean Squared Error MSE2 =", MSE2)
```

```
Mean Squared Error MSE2 = 7.8669534504467356
```

■ **Example (4)**. Compare results by plotting



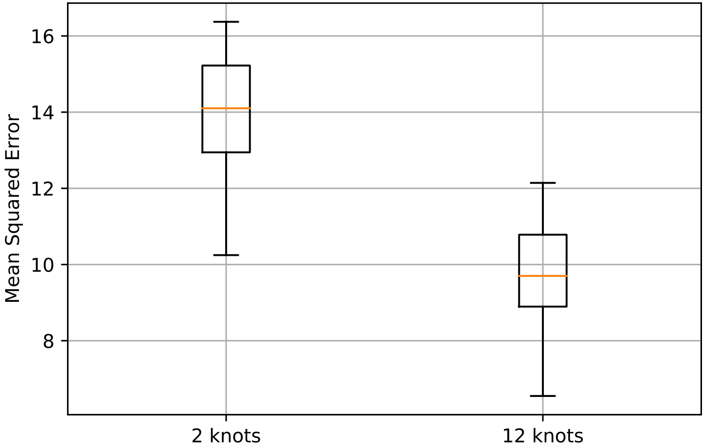
- **Example (5)**. Repeat multiple splits.

```
nshakes=25; MSE1S=[]; MSE2S=[]
for j in range(nshakes):
    XTRAIN,YTRAIN,XTEST,YTEST=shakeupdata(X,Y)
    fit1=LS(XTRAIN,YTRAIN,[2,6,8])
    fit2=LS(XTRAIN,YTRAIN,np.arange(1,12,1))

    YP1=fit1(XTEST)
    YP2=fit2(XTEST)

    MSE1S.append(mean_squared_error(YP1,YTEST))
    MSE2S.append(mean_squared_error(YP2,YTEST))
plt.boxplot([MSE1S,MSE2S]);
plt.xticks([1,2],["2 knots","12 knots"])
plt.grid()
plt.ylabel("Mean Squared Error")
```

■ **Example (6)**. Boxplot of Least Squares Splines



- **Example (1)**: Univariate Spline: Depends on a single parameter

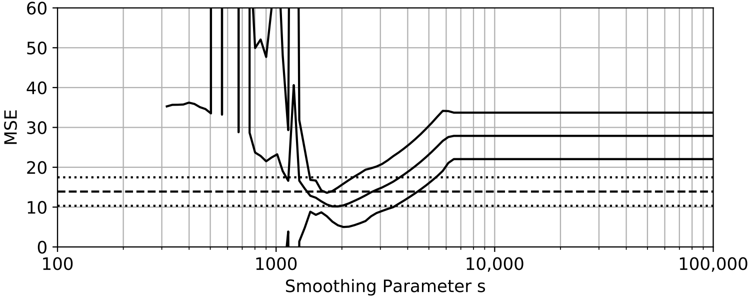
```
from scipy.interpolate import UnivariateSpline as US

def shakeupsim(X,Y):
    XTRAIN,YTRAIN,XTEST,YTEST=shakeupdata(X,Y)
    svals=np.logspace(2.5,5,100)
    MSES=[]
    for sval in svals:
        s=US(XTRAIN,YTRAIN,s=sval)
        MSE = mean_squared_error(s(XTEST), YTEST)
        MSES.append(MSE)
    return(svals, MSES)
```

```
nshakes=25; MSES=[]
for j in range(nshakes):
    svals,M=shakeupsim(X,Y)
    MSES.append(M)
```

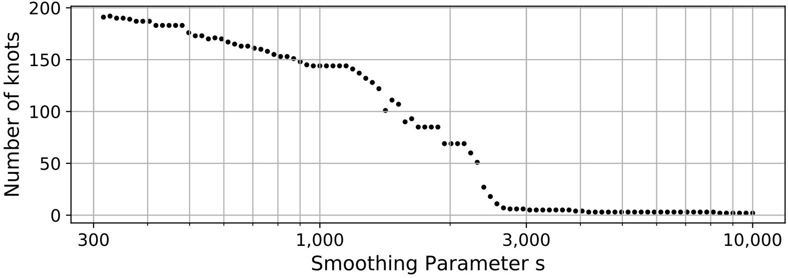
- **Example (2)**. Univariate splines. Mean/Std vs spline parameter.

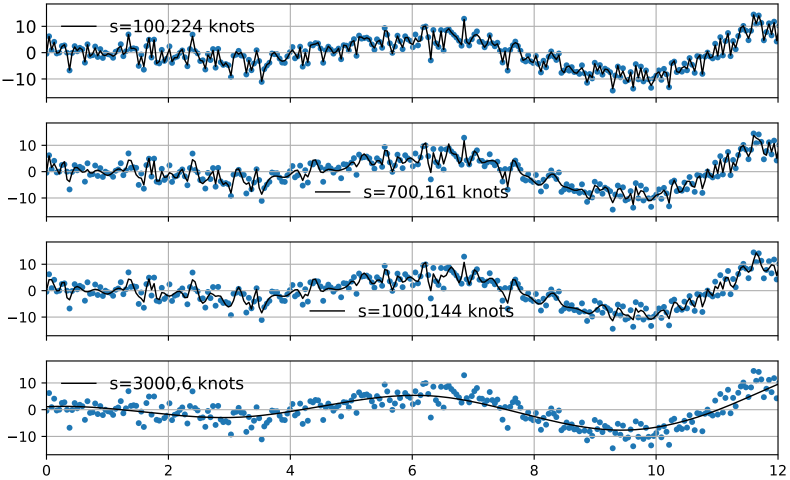
```
MSES = np.array(MSES)
mu=np.mean(MSES,axis=0)
s=np.std(MSES,axis=0)
```



■ **Example (3)**. Spline Parameter. Number of knots.

```
def num_knots(X,Y,spar):
    return len(US(X,Y,s=spar).get_knots())
svals=np.logspace(2.5,4,100)
kvals=[num_knots(X,Y,sval) for sval in svals]
plt.scatter(svals,kvals)
```




s=100,224 knots
s=700,161 knots
s=1000,144 knots
s=3000,6 knots
10
0
−10
0
2
4
6
8
10
12