CMSE 820: Homework #3

Due on September 29, 2019 at 11:59pm

Professor Yuying Xie

Boyao Zhu

Problem 1

1. **Solution**
Start from the truncated power series:

$$f(X) = \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K} \theta_k (X - \xi_k)_+^3$$

For the left boundary knot

$$f(X) = \sum_{j=0}^{3} \beta_j X^j, \qquad X \leq \xi_i$$

and we need the constraints $\beta_2 = 0$ and $\beta_3 = 0$ for the function to be linear.
For the right boundary knot

$$\begin{aligned} f(X) &= \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K} \theta_k (X - \xi_k)_+^3, \qquad X \geq \xi_i \\ &= \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K} \theta_k X^3 - \sum_{k=1}^{K} \theta_k \xi_k 3X^2 + \sum_{k=1}^{K} \theta_k \xi_k^2 3X - \sum_{k=1}^{K} \theta_k \xi_k^3 \end{aligned}$$

and we need the constraints $\theta_k = 0$ and $\sum_{k=1}^{K} \xi_k \theta_k = 0$ for the function to be linear.
Hence, the truncated power series representation

$$f(X) = \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K} \theta_k (X - \xi_k)_+^3$$

with the constraints on the coefficients

$$\beta_2 = 0, \qquad \beta_3 = 0, \qquad \sum_{k=1}^{K} \theta_k = 0, \qquad \sum_{k=1}^{K} \xi_k \theta_k = 0$$

2. **Solution**
Taking into account first the β restrictions, we can construct a new basis with the first two basis function as

$$f(X) = \beta_0 \cdot \underbrace{1}_{N_1(x)} + \beta_1 \cdot \underbrace{X}_{N_2(x)} + 0 \cdot X^2 + 0 \cdot X^3 + \cdots$$

For the θ constraints, we utilize that

$$\sum_{k=1}^{K-2} \theta_k = -\theta_{K-1} - \theta_K, \qquad \sum_{k=1}^{K-2} \xi_k \theta_k = -\xi_{K-1}\theta_{K-1} - \xi_K \theta_K$$

Take out the last two terms of the truncated basis functions:
For the θ constraints, we utilize that

$$\sum_{k=1}^{K} \theta_k (X - \xi_k)_+^3 = \sum_{k=1}^{K-2} \theta_k (X - \xi_k)_+^3 + \theta_{K-1}(X - \xi_{K-1})_+^3 + \theta_K (X - \xi_K)_+^3$$

Problem 1 continued on next page...

Start with the second last term

$$
\begin{aligned}
\theta_{K-1}(X-\xi_{K-1})_+^3 &= \frac{(X-\xi_{K-1})_+^3}{(\xi_K-\xi_{K-1})}\big(\theta_{K-1}(\xi_K-\xi_{K-1})\big)\\
&= \frac{(X-\xi_{K-1})_+^3}{(\xi_K-\xi_{K-1})}\Big(\theta_{K-1}\xi_K-\theta_{K-1}\xi_{K-1}+\underbrace{\theta_K\xi_K-\theta_K\xi_K}_{0}\Big)\\
&= \frac{(X-\xi_{K-1})_+^3}{(\xi_K-\xi_{K-1})}\big(\xi_K(\theta_{K-1}+\theta_K)-\xi_{K-1}\theta_{K-1}-\xi_K\theta_K\big)\\
&= \frac{(X-\xi_{K-1})_+^3}{(\xi_K-\xi_{K-1})}\Big(-\xi_K\sum_{k=1}^{K-2}\theta_k+\sum_{k=1}^{K-2}\theta_k\xi_k\Big) \qquad \text{by constraints}\\
&= -\frac{(X-\xi_{K-1})_+^3}{(\xi_K-\xi_{K-1})}\sum_{k=1}^{K-2}\theta_k(\xi_K-\xi_k)\\
&= -\sum_{k=1}^{K-2}\theta_k(\xi_K-\xi_k)\frac{(X-\xi_{K-1})_+^3}{(\xi_K-\xi_{K-1})}
\end{aligned}
$$

Then, take the last term, and do the same

$$
\begin{aligned}
\theta_K(X-\xi_K)_+^3 &= \frac{(X-\xi_K)_+^3}{(\xi_K-\xi_{K-1})}\Big(\theta_K\xi_K-\theta_K\xi_{K-1}+\underbrace{\theta_{K-1}\xi_{K-1}-\theta_{K-1}\xi_{K-1}}_{0}\Big)\\
&= \frac{(X-\xi_K)_+^3}{(\xi_K-\xi_{K-1})}\big(-\xi_{K-1}(\theta_{K-1}+\theta_K)+\xi_{K-1}\theta_{K-1}+\xi_K\theta_K\big)\\
&= \frac{(X-\xi_K)_+^3}{(\xi_K-\xi_{K-1})}\Big(\xi_{K-1}\sum_{k=1}^{K-2}\theta_k-\sum_{k=1}^{K-2}\theta_k\xi_k\Big) \qquad \text{by constraints}\\
&= (X-\xi_K)_+^3\sum_{k=1}^{K-2}\theta_k\frac{\xi_{K-1}-\xi_k}{(\xi_K-\xi_{K-1})}\\
&= (X-\xi_K)_+^3\sum_{k=1}^{K-2}\theta_k(\xi_K-\xi_k)\frac{\xi_{K-1}-\xi_k+\xi_K-\xi_K}{(\xi_K-\xi_{K-1})(\xi_K-\xi_k)}\\
&= (X-\xi_K)_+^3\sum_{k=1}^{K-2}\theta_k(\xi_K-\xi_k)\Big(\frac{1}{\xi_K-\xi_{K-1}}-\frac{1}{\xi_K-\xi_k}\Big)
\end{aligned}
$$

Problem 1 continued on next page. . .

Then, we combine the two expressions

$$
\begin{aligned}
\sum_{k=1}^{K}\theta_k(X-\xi_k)_+^3 &= \sum_{k=1}^{K-2}\theta_k(X-\xi_k)_+^3 + \theta_{K-1}(X-\xi_{K-1})_+^3 + \theta_K(X-\xi_K)_+^3 \\
&= \sum_{k=1}^{K-2}\theta_k(X-\xi_k)_+^3 - \sum_{k=1}^{K-2}\theta_k(\xi_K-\xi_k)\frac{(X-\xi_{K-1})_+^3}{(\xi_K-\xi_{K-1})} \\
&+ (X-\xi_K)_+^3\sum_{k=1}^{K-2}\theta_k(\xi_K-\xi_k)\Big(\frac{1}{\xi_K-\xi_{K-1}}-\frac{1}{\xi_K-\xi_k}\Big) \\
&= \sum_{k=1}^{K-2}\theta_k(\xi_K-\xi_k)\frac{(X-\xi_k)_+^3}{\xi_K-\xi_k} - \sum_{k=1}^{K-2}\theta_k(\xi_K-\xi_k)\frac{(X-\xi_{K-1})_+^3}{(\xi_K-\xi_{K-1})} \\
&+ \sum_{k=1}^{K-2}\theta_k(\xi_K-\xi_k)\Big(\frac{(X-\xi_K)_+^3}{\xi_K-\xi_{K-1}}-\frac{(X-\xi_K)_+^3}{\xi_K-\xi_k}\Big) \\
&= \sum_{k=1}^{K-2}\theta_k(\xi_K-\xi_k)\Big(\frac{(X-\xi_k)_+^3}{\xi_K-\xi_k}-\frac{(X-\xi_{K-1})_+^3}{(\xi_K-\xi_{K-1})}+\frac{(X-\xi_K)_+^3}{\xi_K-\xi_{K-1}}-\frac{(X-\xi_K)_+^3}{\xi_K-\xi_k}\Big) \\
&= \sum_{k=1}^{K-2}\theta_k(\xi_K-\xi_k)\Big(\frac{(X-\xi_k)_+^3-(X-\xi_K)_+^3}{\xi_K-\xi_k}-\frac{(X-\xi_{K-1})_+^3-(X-\xi_K)_+^3}{\xi_K-\xi_{K-1}}
\end{aligned}
$$

Therefore,

$$
\sum_{k=1}^{K}\theta_k(X-\xi_k)_+^3 = \sum_{k=1}^{K-2}\theta_k(\xi_K-\xi_k)(d_k(X)-d_{K-1}(X))
$$

where

$$
N_{k+2}(X)=d_k(X)-d_{K-1}(X), \qquad d_k(X)=\frac{(X-\xi_k)_+^3-(X-\xi_K)_+^3}{\xi_K-\xi_k}
$$

3. **Solution**
It is easy to verify that the second derivative of basis functions exist and equal 0 at both $\xi_1$and ξ_K. Thus these basis functions satisfy the requirement of Natural Cubic spline.

Problem 2

2. **Solution**
At first, $x_0=a$, $x_{N+1}=b$, and $f_i=f^{'''}(x)$. And for $x\in[x_i,x_{i+1}]$. Also we have $h(x_i)=\tilde{f}(x_i)-f(x_i)=y_i-y_i=0$

$$
\begin{aligned}
\int_a^b f^{''}(x)h^{''}(x)dx &= f^{''}(x)h^{'}(x)|_a^b - \int_a^b f^{'''}(x)h^{'}(x)dx \\
&= \sum_{i=0}^{N}\int_{x_i}^{x_{i+1}} f^{'''}(x)h^{'}(x)dx \qquad (f^{''}(a)=f^{''}(b)=0) \\
&= \sum_{i=0}^{N} f_i\int_{x_i}^{x_{i+1}} h^{'}(x)dx \\
&= \sum_{i=0}^{N} f_i[h(x_{i+1})-h(x_i)] \\
&= 0
\end{aligned}
$$

3. **Solution**

$$\begin{aligned}
\int_a^b f^{''}(x)^2 &\le \int_a^b \tilde{f}^{''}(x)^2 dx \\
&\le \int_a^b (f^{''}(x) + h^{''}(x))^2 dx \\
&\le \int_a^b f^{''}(x)^2 + h^{''}(x)^2 + 2f^{''}(x)h^{''}(x)dx \\
&\le \int_a^b f^{''}(x)^2 + h^{''}(x)^2 dx \qquad (by (b))
\end{aligned}$$

This is trivial and the equality holds when $h^{''}(x) = 0$ or $\tilde{f}(X) - f(X)$.

Problem 3

The code is attached in the back. Prediction error for OLS = 30.037394792483873
Prediction error for ridge = 25.856854295473653
Prediction error for lasso = 148.39318958116104
Note that in this case, Ridge outperforms OLS. But Lasso leads to worst performance.

Problem 4

4a
Cubic Spline

$$\begin{aligned}
h_1(X) &= 1 \\
h_2(X) &= X \\
h_3(X) &= X^2 \\
h_4(X) &= X^3 \\
h_5(X) &= (X - \frac{\pi}{4})_+^3 \\
h_6(X) &= (X - \frac{\pi}{2})_+^3 \\
h_7(X) &= (X - \pi)_+^3 \\
h_8(X) &= (X - \frac{4\pi}{2})_+^3 \\
h_9(X) &= (X - \frac{7\pi}{4})_+^3
\end{aligned}$$

Natural Cubic Spline

$$\begin{aligned}
h_1(X) &= 1 \\
h_2(X) &= X \\
h_3(X) &= -\frac{1}{6}(X - \frac{\pi}{4})_+^3 + (X - \frac{3\pi}{2})_+^3 - \frac{5}{6}(X - \frac{7\pi}{4})_+^3 \\
h_4(X) &= -\frac{1}{5}(X - \frac{\pi}{2})_+^3 + (X - \frac{3\pi}{2})_+^3 - \frac{4}{5}(X - \frac{7\pi}{4})_+^3 \\
h_5(X) &= -\frac{1}{3}(X - \pi)_+^3 + (X - \frac{3\pi}{2})_+^3 - \frac{2}{3}(X - \frac{7\pi}{4})_+^3
\end{aligned}$$

Untitled4

September 29, 2019

1 Problem 3

In [8]:
```
import pandas as pd
import numpy as np

Xtrain = pd.read_csv("Q3_X_train.csv")
ytrain = pd.read_csv("Q3_Y_train.csv")
Xtest  = pd.read_csv("Q3_X_test.csv")
ytest  = pd.read_csv("Q3_y_test.csv")

X = Xtrain.as_matrix()
y = ytrain.as_matrix()
Xt = Xtest.as_matrix()
yt = ytest.as_matrix()

ones = np.ones(500).reshape((500,1))
X = np.hstack((ones,X))
ones = np.ones(250).reshape((250,1))
Xt= np.hstack((ones,Xt))


# Ordinary Least Square
k1 = np.linalg.inv(np.matmul(X.T,X)).dot(X.T).dot(y)
y_pred = Xt.dot(k1)
pred_error = np.mean((yt-y_pred)**2)
print ("Prediction error for ols = ",pred_error)


# Ridge regression
from sklearn.model_selection import KFold
kfold = KFold(5,True,1)
u = np.mean(X,axis=0)
X_cent0 = X-u
I = np.identity(51)
a = [1e-15, 1e-10, 1e-8, 1e-4, 1e-3, 1e-2, 1]
b = np.linspace(2,500)
alpha = np.hstack((a,b))
```

```
data = np.hstack((X_cent0,y))

Error = []
for i in range(len(alpha)):
    pred_error = 0
    for train, test in kfold.split(data):
        X_cent = data[train][:,:-1]
        y_cent = data[train][:,-1]
        X_vald = data[test][:,:-1]
        y_vald = data[test][:,-1]
        k2 = np.linalg.inv(np.matmul(X_cent.T,X_cent)+alpha[i]*I).dot(X_cent.T).dot(y_c
        y_pred = X_vald.dot(k2)
        pred_error += np.mean((y_vald-y_pred)**2)

    pred_error1 = pred_error/5
    Error.append(pred_error1)
import matplotlib.pyplot as plt
#plt.plot(alpha,Error)
#plt.show()

optAlpha = alpha[Error.index(np.min(Error))]

k2 = np.linalg.inv(np.matmul(X_cent0.T,X_cent0)+optAlpha*I).dot(X_cent0.T).dot(y)
y_pred = Xt.dot(k2)
pred_error = np.mean((yt-y_pred)**2)
print ("Prediction error for ridge = ",pred_error)



# Lasso Regression

from sklearn.linear_model import LassoCV, Lasso

Xtrain = pd.read_csv("Q3_X_train.csv")
ytrain = pd.read_csv("Q3_Y_train.csv")
Xtest  = pd.read_csv("Q3_X_test.csv")
ytest  = pd.read_csv("Q3_y_test.csv")

X = Xtrain.as_matrix()
y = ytrain.as_matrix()
Xt = Xtest.as_matrix()
yt = ytest.as_matrix()

ones = np.ones(500).reshape((500,1))
X = np.hstack((ones,X))
ones = np.ones(250).reshape((250,1))
Xt= np.hstack((ones,Xt))
```

```
        u = np.mean(X,axis=0)
        X = X-u

        model = LassoCV(cv=5,max_iter=3000)
        model.fit(X,y)
        ynew = model.predict(Xt)
        pred_error = np.mean((yt-ynew)**2)
        print ("Prediction error for lasso = ",pred_error)
```

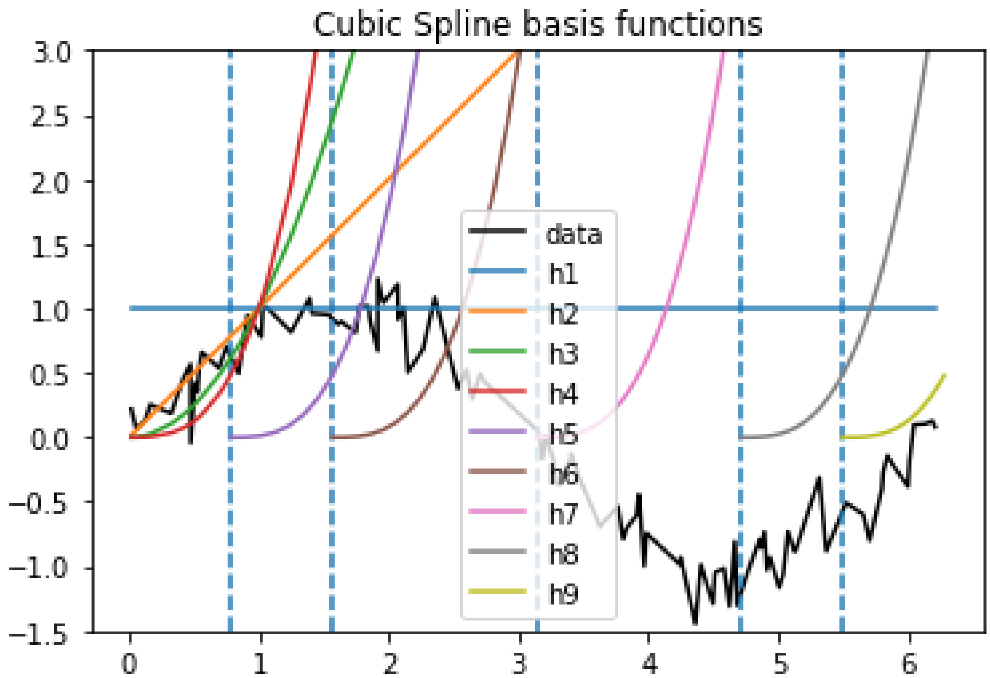
```
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:9: FutureWarning: Method .as_matr
  if __name__ == '__main__':
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:10: FutureWarning: Method .as_mat
  # Remove the CWD from sys.path while we load stuff.
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:11: FutureWarning: Method .as_mat
  # This is added back by InteractiveShellApp.init_path()
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:12: FutureWarning: Method .as_mat
  if sys.path[0] == '':
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:74: FutureWarning: Method .as_mat
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:75: FutureWarning: Method .as_mat
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:76: FutureWarning: Method .as_mat
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:77: FutureWarning: Method .as_mat
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/coordinate_descent.py:1109: DataCo
  y = column_or_1d(y, warn=True)


Prediction error for ols =  30.037394792483873
Prediction error for ridge =  25.856854295473653
Prediction error for lasso =  148.39318958116104
```

2 Problem 4a

```
In [287]: import numpy as np
          import matplotlib.pyplot as plt
          xi1 = np.pi/4
          xi2 = np.pi/2
          xi3 = np.pi
          xi4 = np.pi*3/2
          xi5 = np.pi*7/4

          X = pd.read_csv("Q4_X.csv").as_matrix().reshape(-1)
          y = pd.read_csv("Q4_Y.csv").as_matrix().reshape(-1)
          plt.plot(X,y,'k',label="data")
          plt.axvline(x=xi1,linestyle="--")
          plt.axvline(x=xi2,linestyle="--")
          plt.axvline(x=xi3,linestyle="--")
          plt.axvline(x=xi4,linestyle="--")
          plt.axvline(x=xi5,linestyle="--")
```

```
h1 = np.ones(len(X))
plt.plot(X,h1,label="h1")
h2 = X
plt.plot(X,h2,label="h2")
h3 = X**2
plt.plot(X,h3,label="h3")
h4 = X**3
plt.plot(X,h4,label="h4")

X5 = np.linspace(xi1,6.28)
h5 = (X5-xi1)**3
plt.plot(X5,h5,label="h5")

X6 = np.linspace(xi2,6.28)
h6 = (X6-xi2)**3
plt.plot(X6,h6,label="h6")

X7 = np.linspace(xi3,6.28)
h7 = (X7-xi3)**3
plt.plot(X7,h7,label="h7")

X8 = np.linspace(xi4,6.28)
h8 = (X8-xi4)**3
plt.plot(X8,h8,label="h8")

X9 = np.linspace(xi5,6.28)
h9 = (X9-xi5)**3
plt.plot(X9,h9,label="h9")

plt.ylim(-1.5,3)
plt.legend()
plt.title("Cubic Spline basis functions")
plt.show()
```

```
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:9: FutureWarning: Method .as_matri
  if __name__ == '__main__':
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:10: FutureWarning: Method .as_matr
  # Remove the CWD from sys.path while we load stuff.
```

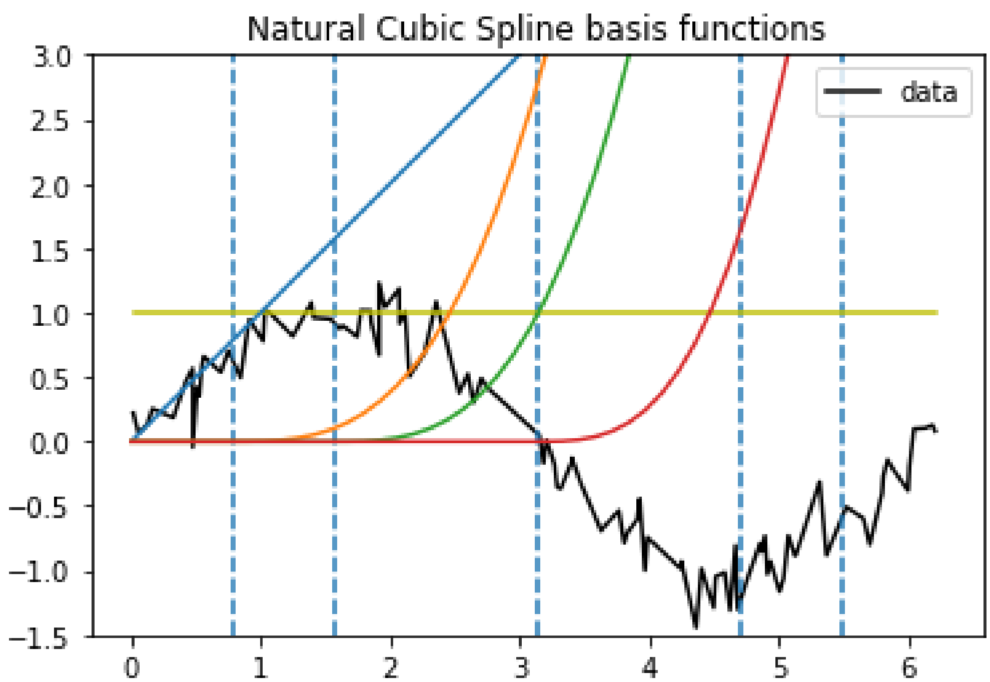


3 Problem 4b

```
In [319]: import numpy as np
          import matplotlib.pyplot as plt
          xi1 = np.pi/4
          xi2 = np.pi/2
          xi3 = np.pi
          xi4 = np.pi*3/2
          xi5 = np.pi*7/4

          X = pd.read_csv("Q4_X.csv").as_matrix().reshape(-1)
          y = pd.read_csv("Q4_Y.csv").as_matrix().reshape(-1)
          plt.plot(X,y,'k',label="data")
          plt.axvline(x=xi1,linestyle="--")
          plt.axvline(x=xi2,linestyle="--")
          plt.axvline(x=xi3,linestyle="--")
          plt.axvline(x=xi4,linestyle="--")
          plt.axvline(x=xi5,linestyle="--")

          h1 = np.ones(len(X))
          plt.plot(X,h1,'y')
          h2 = X
          plt.plot(X,h2)
```

```
x = np.linspace(0,6.28)
b1 = np.piecewise(x,[x<xi1,x>=xi1],[lambda x:0,lambda x: (x-xi1)**3])
b2 = np.piecewise(x,[x<xi2,x>=xi2],[lambda x:0,lambda x: (x-xi2)**3])
b3 = np.piecewise(x,[x<xi3,x>=xi3],[lambda x:0,lambda x: (x-xi3)**3])
b4 = np.piecewise(x,[x<xi4,x>=xi4],[lambda x:0,lambda x: (x-xi4)**3])
b5 = np.piecewise(x,[x<xi5,x>=xi5],[lambda x:0,lambda x: (x-xi5)**3])

d1 = (b1-b5)/(xi5-xi1)
d2 = (b2-b5)/(xi5-xi2)
d3 = (b3-b5)/(xi5-xi3)
d4 = (b4-b5)/(xi5-xi4)

h3 = d1-d4
plt.plot(x,h3)
h4 = d2-d4
plt.plot(x,h4)
h5 = d3-d4
plt.plot(x,h5)



plt.legend()
plt.ylim(-1.5,3)
plt.title("Natural Cubic Spline basis functions")
plt.show()
```

```
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:9: FutureWarning: Method .as_matri
  if __name__ == '__main__':
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:10: FutureWarning: Method .as_matr
  # Remove the CWD from sys.path while we load stuff.
```




```
In [318]: X = pd.read_csv("Q4_X.csv").as_matrix().reshape(-1)
          y = pd.read_csv("Q4_Y.csv").as_matrix().reshape(-1)
          plt.plot(X,y,'k',label="data")
          y_ture = np.sin(X)
          plt.plot(X,y_ture,'rs',label="true")

          from scipy.interpolate import CubicSpline, interp1d
          import matplotlib.pyplot as plt
          import numpy as np

          xnew = np.linspace(0,2*np.pi,num=7,endpoint=True)
          y1 = np.sin(xnew)

          f = interp1d(xnew,y1,kind='cubic')
          xxnew = np.linspace(-1,7,num=100)
          #plt.plot(xxnew,f(xxnew),label="cubic")
          cs = CubicSpline(xnew,y1,extrapolate=True)
          plt.plot(xxnew,cs(xxnew),'g',label='cubic')

          cs = CubicSpline(xnew,y1,bc_type='natural',extrapolate=True)
          plt.plot(xxnew,cs(xxnew),'y',label='natural')

          #plt.plot(xnew,y1,'ko')

          plt.axvline(x=xi1,linestyle="-.")
```

```
plt.axvline(x=xi2,linestyle="-.")
plt.axvline(x=xi3,linestyle="-.")
plt.axvline(x=xi4,linestyle="-.")
plt.axvline(x=xi5,linestyle="-.")

plt.title("Natural Cubic Spline fitting")
plt.legend()
plt.show()
```

```
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: FutureWarning: Method .as_matr
  """Entry point for launching an IPython kernel.
/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2: FutureWarning: Method .as_matr
```