Lecture 3 part 4

July 17, 2021

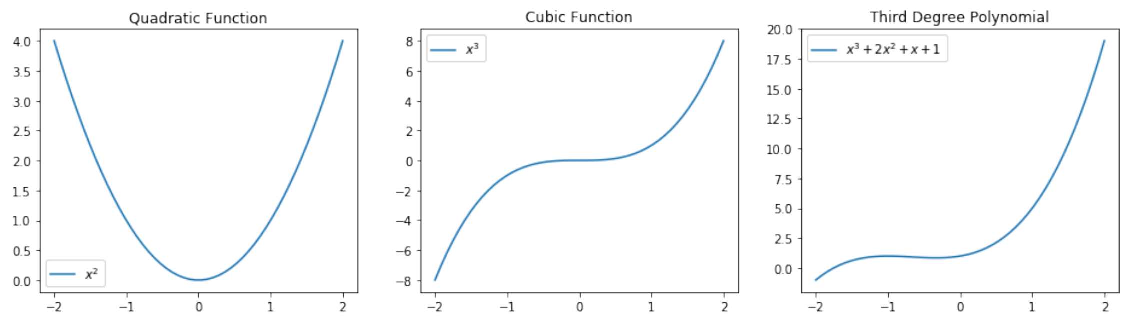
[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
import warnings
```

1 Quadratic functions

[2]:
```
warnings.filterwarnings('ignore')
plt.rcParams['figure.figsize']=[16,4]
x_vars = np.linspace(-2,2)
```

[3]:
```
plt.subplot('131')
plt.title ('Quadratic Function')
plt.plot(x_vars, x_vars**2)
plt.legend(['$x^2$'])

plt.subplot('132')
plt.title ('Cubic Function')
plt.plot(x_vars, x_vars**3)
plt.legend(['$x^3$'])

plt.subplot('133')
plt.title ('Third Degree Polynomial')
plt.plot(x_vars, x_vars**3 + 2*x_vars**2 + x_vars+1)
plt.legend(['$x^3 + 2x^2 + x + 1$'])
```

[3]:
```
<matplotlib.legend.Legend at 0x7f58200cd2b0>
```



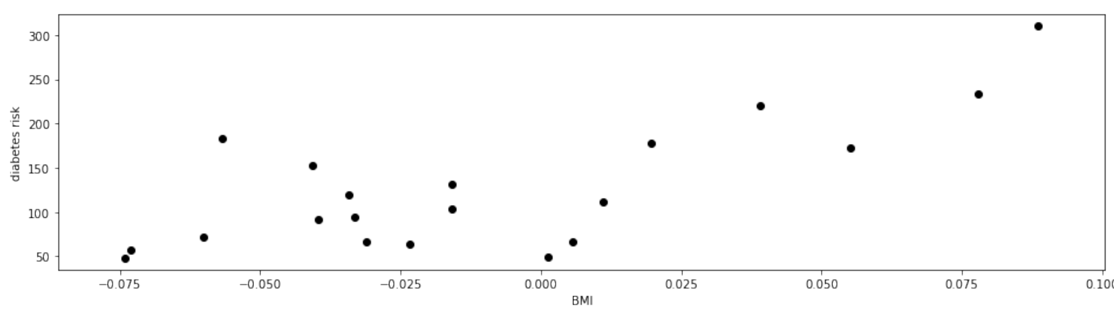
2 UCI DIABETES DATASET

[4]:
```python
X, y = datasets.load_diabetes(return_X_y=True, as_frame=True)
```

[5]:
```python
X['one']= 1
```

[6]:
```python
X_train = X.iloc[-20:]
y_train = y.iloc[-20:]

plt.scatter(X_train.bmi, y_train, color='black')
plt.xlabel('BMI')
plt.ylabel('diabetes risk')
```

[6]:
```
Text(0, 0.5, 'diabetes risk')
```



3 Non linear featurization

[7]:
```python
#X_bmi is a design matrix with just one single feature, bmi

X_bmi = X_train.bmi

X_bmi_p3 = pd.concat([X_bmi, X_bmi**2, X_bmi**3], axis =1)
X_bmi_p3.columns= ['bmi', 'bmi2', 'bmi3']
```

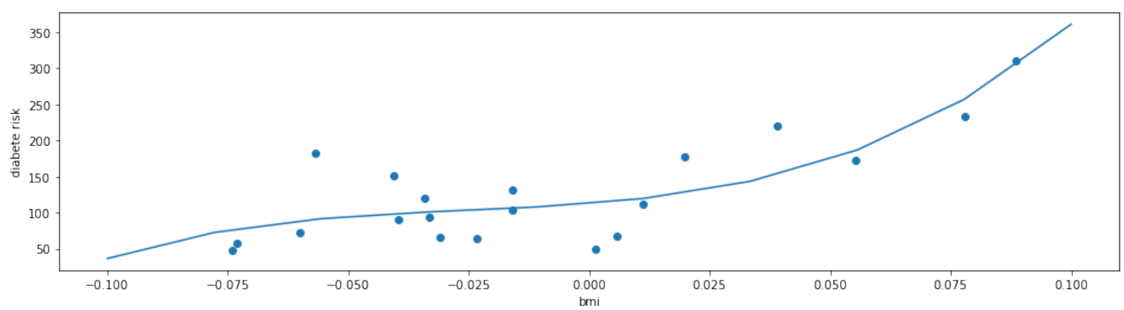
```
X_bmi_p3['one'] = 1
X_bmi_p3.head()
```

[7]:

	bmi	bmi2	bmi3	one
422	0.077863	0.006063	0.000472	1
423	-0.039618	0.001570	-0.000062	1
424	0.011039	0.000122	0.000001	1
425	-0.040696	0.001656	-0.000067	1
426	-0.034229	0.001172	-0.000040	1

[15]:
```
#Fit the linear regression
#This equation comes from the gradient of MSE = 0
theta = np.linalg.inv(X_bmi_p3.T.dot(X_bmi_p3)).dot(X_bmi_p3.T).dot(y_train)

#show the learned polynomial curve
x_line = np.linspace(-0.1,0.1,10)
x_line_p3 = np.stack([x_line, x_line**2, x_line**3, np.ones(10,)], axis=1)
y_train_pred = x_line_p3.dot(theta)

plt.xlabel('bmi')
plt.ylabel('diabete risk')
plt.scatter (X_bmi,y_train)
plt.plot(x_line,y_train_pred)
```

[15]: [<matplotlib.lines.Line2D at 0x7f582007a400>]



[27]:
```
y_train.shape
```

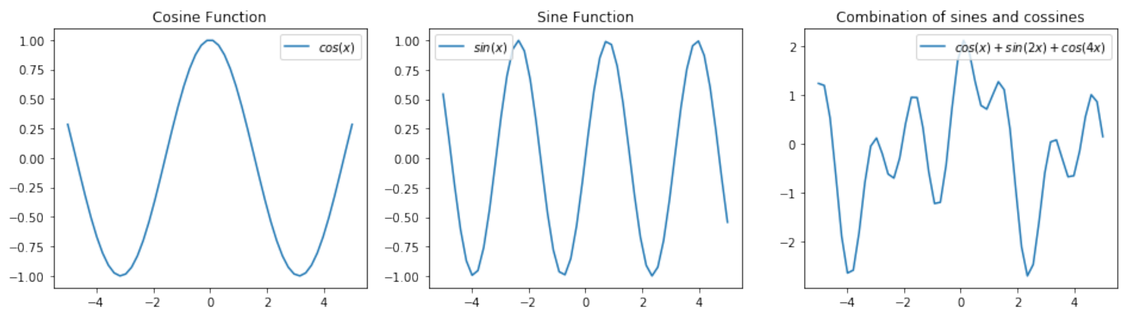
[27]: (20,)

4 Trigonometric functions

[28]:
```
x_vars = np.linspace(-5,5)


plt.subplot('131')
plt.title('Cosine Function')
```

```
plt.plot(x_vars, np.cos(x_vars))
plt.legend(['$cos(x)$'])

plt.subplot('132')
plt.title('Sine Function')
plt.plot(x_vars, np.sin(2*x_vars))
plt.legend(['$sin(x)$'])

plt.subplot('133')
plt.title('Combination of sines and cossines')
plt.plot(x_vars, np.cos(x_vars)+np.sin(2*x_vars)+np.cos(4*x_vars))
plt.legend(['$cos(x)+sin(2x)+cos(4x)$'])
```

[28]:
```
<matplotlib.legend.Legend at 0x7f57c02574e0>
```



5 Fitting BMI with trigonometric functions

[29]:
```
#X_bmi is a design matrix with just one single feature, bmi

X_bmi_p3 = pd.concat([X_bmi, np.cos(X_bmi), np.sin(X_bmi)], axis =1)
X_bmi_p3.columns= ['bmi', 'cos(bmi)', 'sin(bmi)']
X_bmi_p3['one'] = 1
X_bmi_p3.head()
```

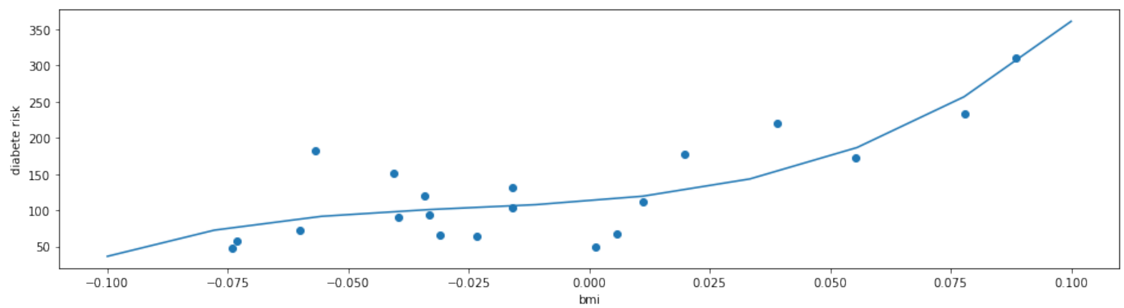
[29]:

	bmi	cos(bmi)	sin(bmi)	one
422	0.077863	0.996970	0.077785	1
423	-0.039618	0.999215	-0.039608	1
424	0.011039	0.999939	0.011039	1
425	-0.040696	0.999172	-0.040685	1
426	-0.034229	0.999414	-0.034222	1

[31]:
```
theta = np.linalg.inv(X_bmi_p3.T.dot(X_bmi_p3)).dot(X_bmi_p3.T).dot(y_train)

#show the learned polynomial curve
x_line = np.linspace(-0.1,0.1,10)
```

```
x_line_p3 = np.stack([x_line, np.cos(x_line), np.sin(x_line), np.ones(10,)],␣
 ↪axis=1)
y_train_pred = x_line_p3.dot(theta)

plt.xlabel('bmi')
plt.ylabel('diabete risk')
plt.scatter (X_bmi,y_train)
plt.plot(x_line,y_train_pred)
```

[31]: [<matplotlib.lines.Line2D at 0x7f57c0346128>]



[ ]: