In [ ]:
```
from __future__ import division
from __future__ import print_function

import numpy as np
import matplotlib

matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
```

Generating data

In [ ]:
```
np.random.seed(0)  # set the random seed for reproducibility

def generate_sample(xmin, xmax, sample_size):
    x = np.linspace(start=xmin, stop=xmax, num=sample_size)
    pix = np.pi * x
    target = np.sin(pix) / pix + 0.1 * x
    noise = 0.05 * np.random.normal(loc=0., scale=1., size=sample_size)
    return x, target, target + noise


def calc_design_matrix(x, c, h):
    return np.exp(-(x[None] - c[:, None]) ** 2 / (2 * h ** 2))
```

In [ ]:
```
# create sample
sample_size = 50
xmin, xmax = -3, 3
x, ytrue, y = generate_sample(xmin=xmin, xmax=xmax, sample_size=sample_size)
```

Cross Validation

In [ ]:
```
def cross_validation(x,y,lm,hh,sample_size):
    # Erros list
    errors = np.zeros((hh.size,lm.size,))
    # Theta list
    thetas = []

    # loop over the badwidths h
    for n1,h in enumerate(hh):
        # loop over the lambda values l
        for n2,l in enumerate(lm):
            # loop over the dataset leaving out one sample at each iteration
            err = 0 # Store test errors
            for i in range(sample_size):

                # Compute cross validation training and validation data
                x_val = np.atleast_1d(x[i])
                y_val = y[i]
                x_loocv = np.delete(x,i)
                y_loocv = np.delete(y,i)
                # calculate design matrix
                k = calc_design_matrix(x_loocv, x_loocv, h)
                # Solve the least square problem
                theta = np.linalg.solve(
                    k.T.dot(k) + l * np.identity(len(k)),
                    k.T.dot(y_loocv[:, None]))
                # Compute prediction
                K = calc_design_matrix(x_loocv, x_val, h)
                prediction = K.dot(theta)
                # Compute squared error and store
                err+=(np.ndarray.item(prediction)-y[i])**2
            # Store mean errors for different parameter values.
            errors[n1,n2]=(err/sample_size)
            # Store the learned parameter
            thetas.append(theta)

    min_in = np.unravel_index(errors.argmin(), errors.shape)
    print(f"Minimum Cross Validation Error is {errors[min_in]} at Lambda = {lm[min_in[1]]} and Bandwidth
```

```python
    return thetas,errors
```

Run Cross Validation

In [ ]:

```python
#define lambda values
lm = np.array([0.0001,0.1,100])
#define range of gaussian bandwidth h
hh = np.array([0.03,0.3,3])

thetas,errors = cross_validation(x,y,lm,hh,sample_size)
```

```
Minimum Cross Validation Error is 0.0037134179245161066 at Lambda = 0.1 and Bandwidth = 0.3
```

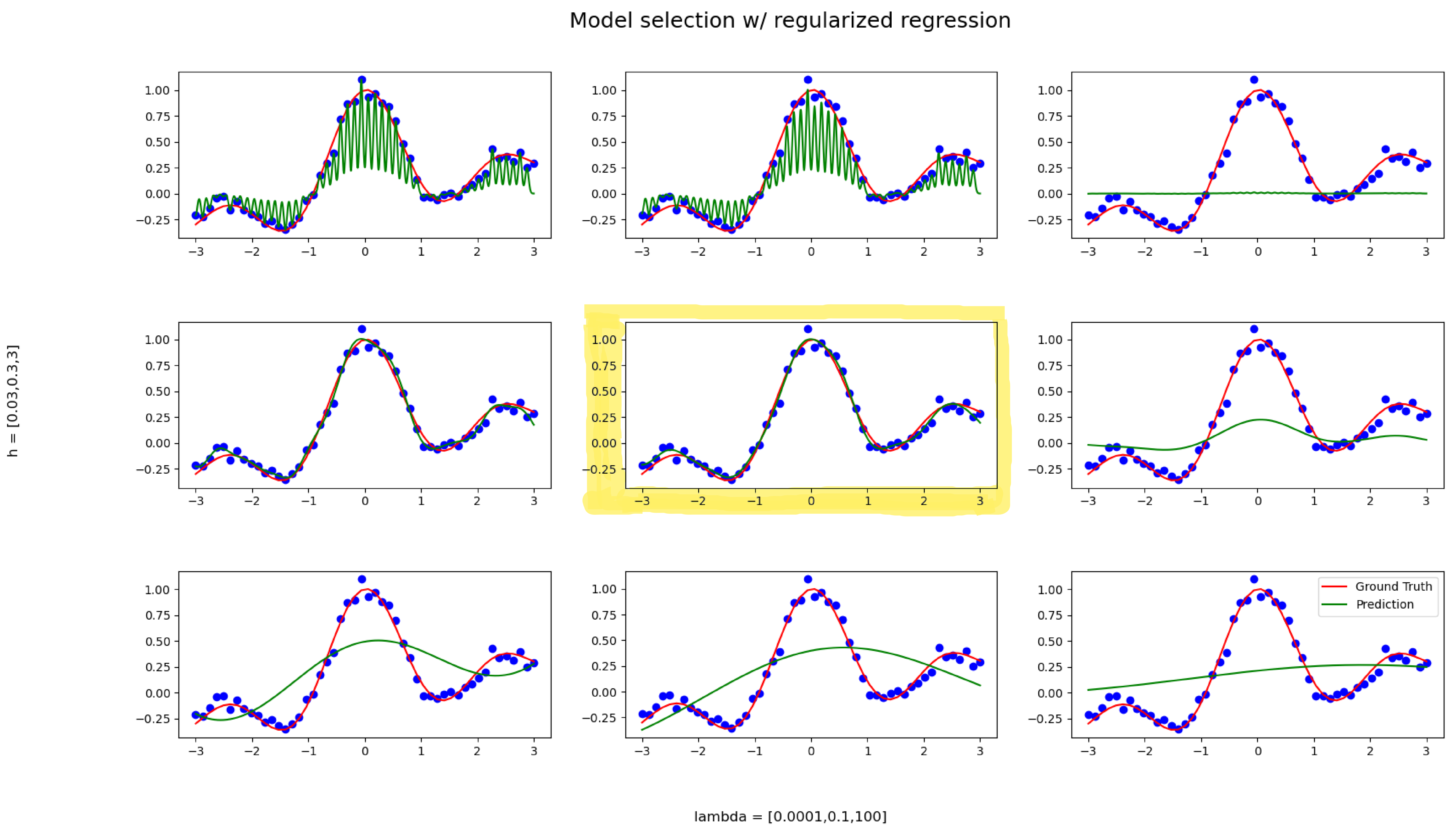
Plotting

In [ ]:

```python
# create data to visualize the prediction
X = np.linspace(start=xmin, stop=xmax, num=5000)

# define subplot grid
fig, axs = plt.subplots(nrows=3, ncols=3, figsize=(20, 20))
plt.subplots_adjust(hspace=0.5)
fig.suptitle("Model selection w/ regularized regression", fontsize=18, y=0.95)

comb_array = np.array(np.meshgrid(hh, lm)).T.reshape(-1, 2)
```

In [ ]:

```python
for theta,c,ax in zip(thetas,comb_array,axs.ravel()):
    #print(theta,l,h)
    h,l = c[0],c[1]
    K = calc_design_matrix(x[:-1], X, h)
    prediction = K.dot(theta)
    # visualization
    ax.scatter(x, y, c='blue', marker='o')
    ax.plot(x, ytrue, c='red',label="Ground Truth")
    ax.plot(X, prediction, c="green", label="Prediction")

    plt.legend()
    fig.supxlabel('lambda = [0.0001,0.1,100]')
    fig.supylabel('h = [0.03,0.3,3]')

plt.show()
#plt.savefig('output.png')
```


Model selection w/ regularized regression
h = [0.03,0.3,3]
lambda = [0.0001,0.1,100]
Ground Truth
Prediction
1.00
0.75
0.50
0.25
0.00
−0.25
−3
−2
−1
0
1
2
3

Advanced Data Analysis
Homework Week 2

Aswin Vijay

April 24, 2023

Homework 2

The Mean Squared Error for Leave one out Cross Validation is given by,

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{f}_i(x_i) - y_i\right)^2$$

where $\hat{f}_i(x)$ is learned from data other than (x_i, y_i). The given linear model is,

$$f_\theta(x) = \sum_{j=1}^{b}\theta_j\phi(x)_j = \phi^T\theta$$

Calculating optimal $\hat{\theta}_i$ learned by removing (x_i, y_i) we get the optimization objective as,

$$\hat{\theta}_i = \arg\min_\theta\left[\frac{1}{2}\left(\sum_{k=1}^{n}(f_\theta(x_k) - y_k)^2 - (f_\theta(x_i) - y_i)^2\right) + \frac{\lambda}{2}||\theta||^2\right]$$

Using $\Phi, \mathbf{y}, \theta, \phi_i, y_i$ we get the regularized objective as,

$$\hat{\theta}_i = \arg\min_\theta\left[\frac{1}{2}||\Phi\theta - \mathbf{y}||^2 - \frac{1}{2}||\phi_i\theta - y_i||^2 + \frac{\lambda}{2}||\theta||^2\right]$$

Taking the derivative and setting to zero we get,

$$\nabla_\theta\left[\frac{1}{2}||\Phi\theta - \mathbf{y}||^2 - \frac{1}{2}||\phi_i\theta - y_i||^2 + \frac{\lambda}{2}||\theta||^2\right] = 0$$

$$\Phi^T\Phi\theta - \Phi^T\mathbf{y} - \phi_i^T\phi_i\theta + \phi_i y_i + \lambda\theta = 0$$

$$\hat{\theta}_i = \frac{\Phi^T\mathbf{y} - \phi_i y_i}{\Phi^T\Phi \ - \phi_i^T\phi_i + \lambda I}$$

Where, Setting $U = \Phi^T\Phi \ + \lambda I$ we get,

$$\hat{\theta}_i = \frac{\Phi^T\mathbf{y} - \phi_i y_i}{U - \phi_i^T\phi_i}$$

Calculating the MSE using prediction for y_i as $\phi_i^T\hat{\theta}_i$ and summing the squared error we get,

$$\begin{aligned} MSE &= \frac{1}{n}\sum_{i=1}^{n}\left(\phi_i^T\hat{\theta}_i - y_i\right)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}\left(\phi_i^T\left[\frac{\Phi^T\mathbf{y} - \phi_i y_i}{U - \phi_i^T\phi_i}\right] - y_i\right)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}\left[(U - \phi_i^T\phi_i)^{-1}(\phi_i^T\Phi^T\mathbf{y} - \phi_i^T\phi_i y_i) - y_i\right]^2 \end{aligned}$$

Using the special case of Sherman-Morrison-Woodbury below we have,

$$\begin{aligned} MSE &= \frac{1}{n}\sum_{i=1}^{n}\left[(U - \phi_i^T\phi_i)^{-1}(\phi_i^T\Phi^T\mathbf{y} - \phi_i^T\phi_i y_i) - y_i\right]^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}\left[(U - \phi_i^T\phi_i)^{-1}(\phi_i^T\Phi^T\mathbf{y} - U y_i)\right]^2 \\ &= \frac{1}{n}||(U - \Phi^T\Phi)^{-1}(\Phi^T\Phi^T\mathbf{y} - U\mathbf{y})||^2 \\ &= \frac{1}{n}||\left([U^{-1} + \frac{U^{-1}\Phi^T\Phi U^{-1}}{I - \Phi U^{-1}\Phi^T}][\Phi^T\Phi^T\mathbf{y} - U\mathbf{y}]\right)||^2 \quad Sherman - Morrison - Woodbury \\ &= \frac{1}{n}||\left(U^{-1}\Phi^T\Phi^T\mathbf{y} - \mathbf{y} + \frac{U^{-1}\Phi^T\Phi U^{-1}\Phi^T\Phi^T\mathbf{y} - U^{-1}\Phi^T\Phi\mathbf{y}}{I - \Phi U^{-1}\Phi^T}\right)||^2 \end{aligned}$$

Next we expand out the denominators and using $H = I - \Phi U^{-1}\Phi^T$ we get the result,

$$MSE = \frac{1}{n}\left\|\frac{H\mathbf{y}}{H}\right\|$$