Linear Regression with Noise and Regularization

Created by Philip Graff for EN.601.475/675 Spring 2020

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression, Lasso, Ridge
```

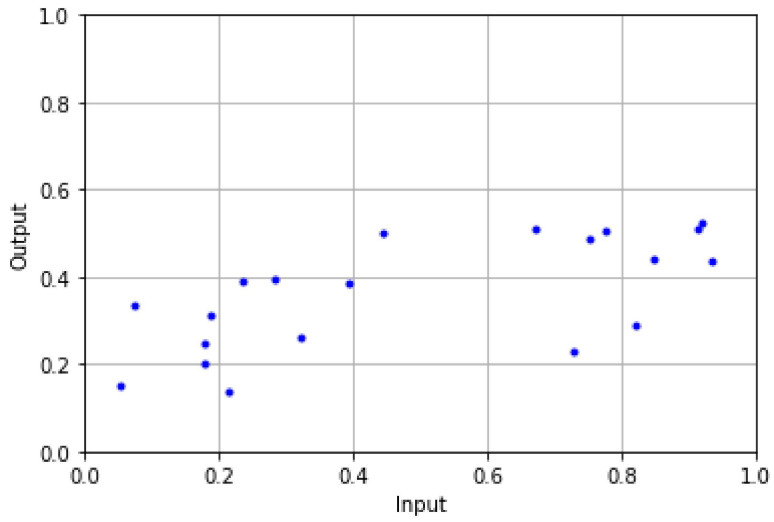
Setup

Set the dataset size as well as true model and noise parameters.

In [2]:
```
N = 20
x = np.random.rand(N,1)
a = 0.2
b = 0.4
sigma = 0.1
y = a + b * x + np.random.randn(N,1)*sigma
```

Plot the Data

In [3]:
```
plt.plot(x, y, '.b')
plt.xlim([0,1])
plt.ylim([0,1])
plt.xlabel('Input')
plt.ylabel('Output')
plt.grid()
plt.show()
```

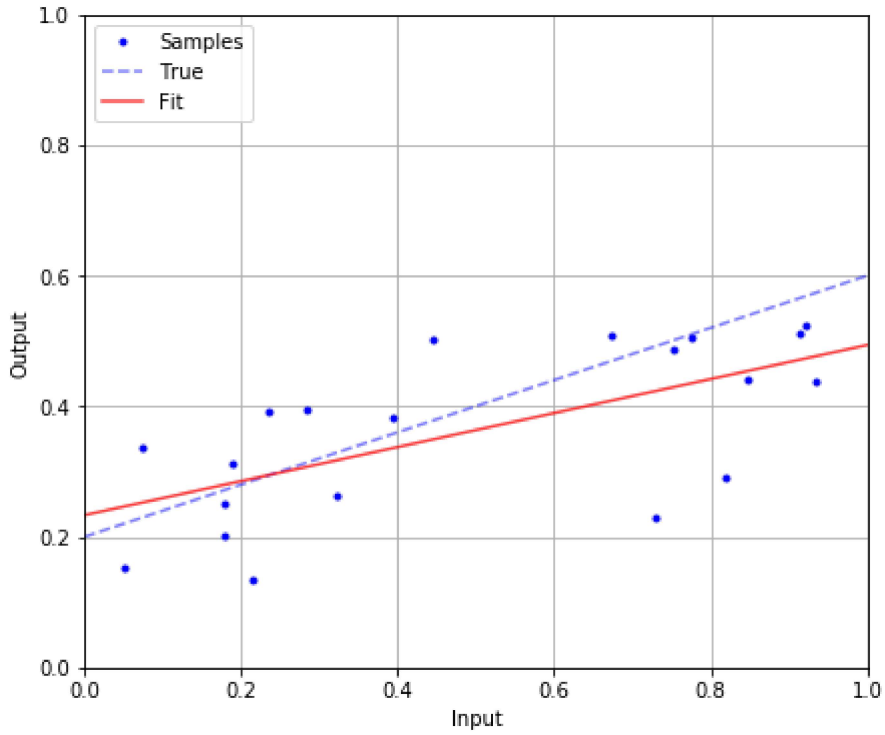


Fit a Model

Fit a regression model without any regularization and plot the fit versus the data and true model.

In [4]:
```
func = LinearRegression()
func.fit(x, y)
xp = np.linspace(0,1,10).reshape(-1,1)
yt = a + b * xp
yp = func.predict(xp)
```

In [5]:
```
plt.figure(figsize=(7,6))
plt.plot(x, y, '.b', label='Samples')
plt.plot(xp, yt, '--b', alpha=0.5, label='True')
plt.plot(xp, yp, '-r', alpha=0.75, label='Fit')
plt.xlim([0,1])
plt.ylim([0,1])
plt.xlabel('Input')
plt.ylabel('Output')
plt.grid()
plt.legend(loc='upper left')
plt.show()
```

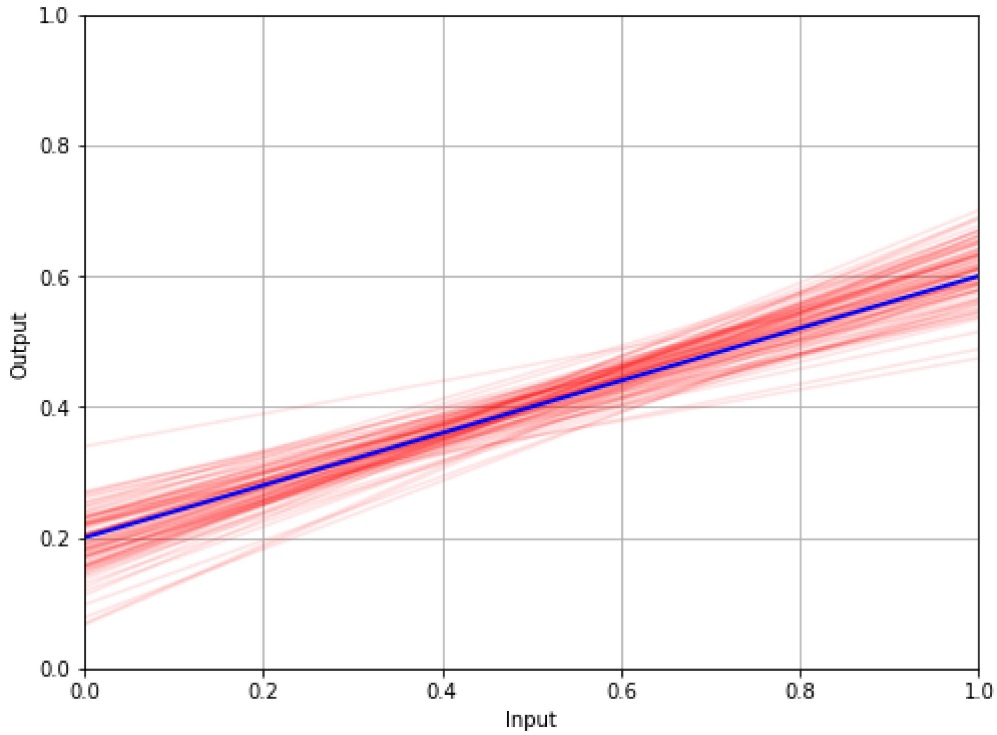


How Stable is this w.r.t. Noise?

Create multiple datasets, each with a different noise realization and see how the fit varies over the different examples of noise.

In [6]:
```
def get_sample_fit(n, sigma, a, b, reg=None, alpha=1.0):
    x = np.random.rand(n,1)
    y = a + b * x + np.random.randn(n,1)*sigma
    if reg is None or reg == 0:
        m = LinearRegression()
        m.fit(x, y)
        return m
    elif reg == 1:
        m = Lasso(alpha=alpha)
        m.fit(x, y)
        return m
    elif reg == 2:
        m = Ridge(alpha=alpha)
        m.fit(x, y)
        return m
    else:
        raise ValueError()
```

In [7]:
```
fig, ax = plt.subplots(1,1,figsize=(8,6))
for _ in range(100):
    func = get_sample_fit(N, sigma, a, b)
    yp = func.predict(xp)
    ax.plot(xp, yp, '-r', alpha=0.1)
ax.plot(xp, yt, '-b', lw=2)
ax.set_xlim([0,1])
ax.set_ylim([0,1])
ax.set_xlabel('Input')
ax.set_ylabel('Output')
ax.grid()
plt.show()
```

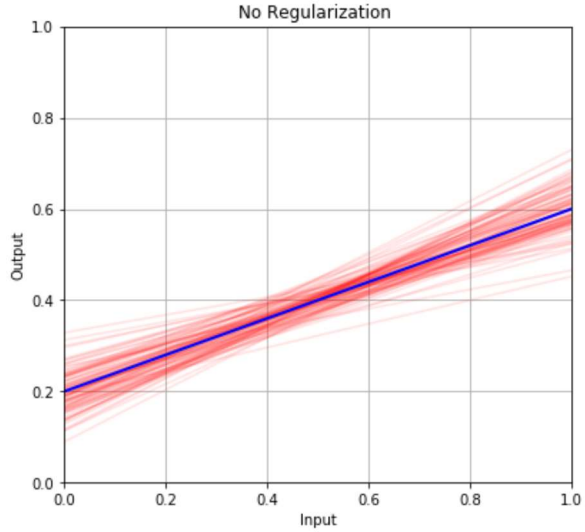


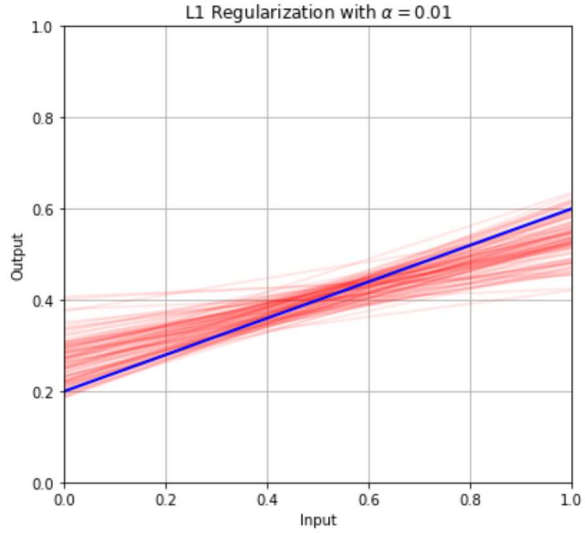
Add Regularization

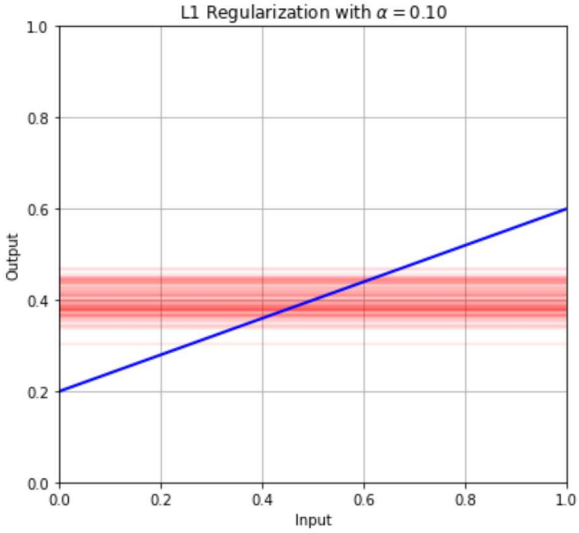
Repeat, but now with L1 and L2 regularization

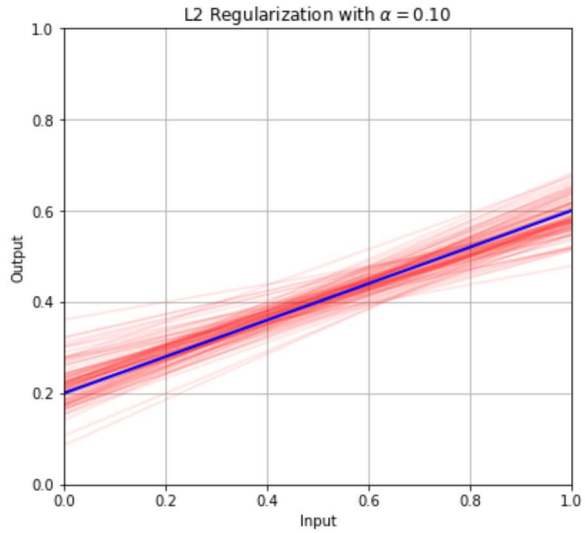
In [8]:

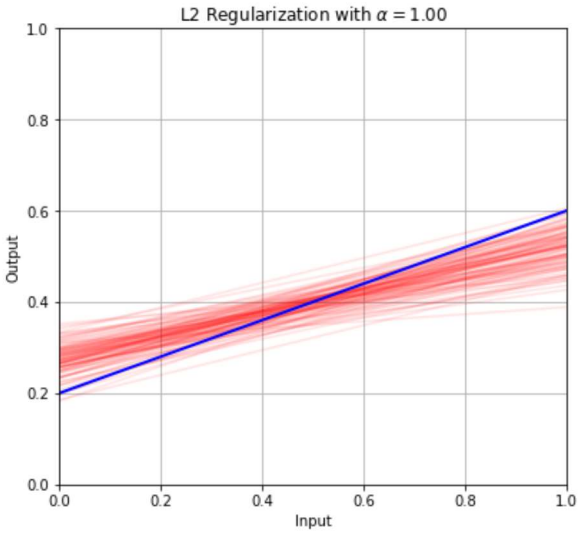
```
alpha_set = [[0.01, 0.1],[0.1,1.0]]
fig, ax = plt.subplots(3,2,figsize=(14,20))
for _ in range(100):
    func = get_sample_fit(N, sigma, a, b, reg=0)
    yp = func.predict(xp)
    ax[0,0].plot(xp, yp, '-r', alpha=0.1)
ax[0,0].plot(xp, yt, '-b', lw=2)
ax[0,0].set_xlim([0,1])
ax[0,0].set_ylim([0,1])
ax[0,0].set_xlabel('Input')
ax[0,0].set_ylabel('Output')
ax[0,0].set_title('No Regularization ')
ax[0,0].grid()
ax[0,1].axis('off')
for i in range(1,3):
    for j in range(2):
        alpha = alpha_set[i-1][j]
        for _ in range(100):
            func = get_sample_fit(N, sigma, a, b, reg=i, alpha=alpha)
            yp = func.predict(xp)
            ax[i,j].plot(xp, yp, '-r', alpha=0.1)
        ax[i,j].plot(xp, yt, '-b', lw=2)
        ax[i,j].set_xlim([0,1])
        ax[i,j].set_ylim([0,1])
        ax[i,j].set_xlabel('Input')
        ax[i,j].set_ylabel('Output')
        ax[i,j].set_title(r'L%d Regularization with $\alpha=%.2f$'%(i,alpha))
        ax[i,j].grid()
plt.show()
```


No Regularization
Output
Input
0.0
0.2
0.4
0.6
0.8
1.0


L1 Regularization with $\alpha = 0.01$
Output
Input
0.0
0.2
0.4
0.6
0.8
1.0


L1 Regularization with $\alpha = 0.10$
Output
Input
0.0
0.2
0.4
0.6
0.8
1.0


L2 Regularization with $\alpha = 0.10$
Output
Input
0.0
0.2
0.4
0.6
0.8
1.0


L2 Regularization with $\alpha = 1.00$
Output
Input
0.0
0.2
0.4
0.6
0.8
1.0

In [ ]: