Lab5

November 7, 2024

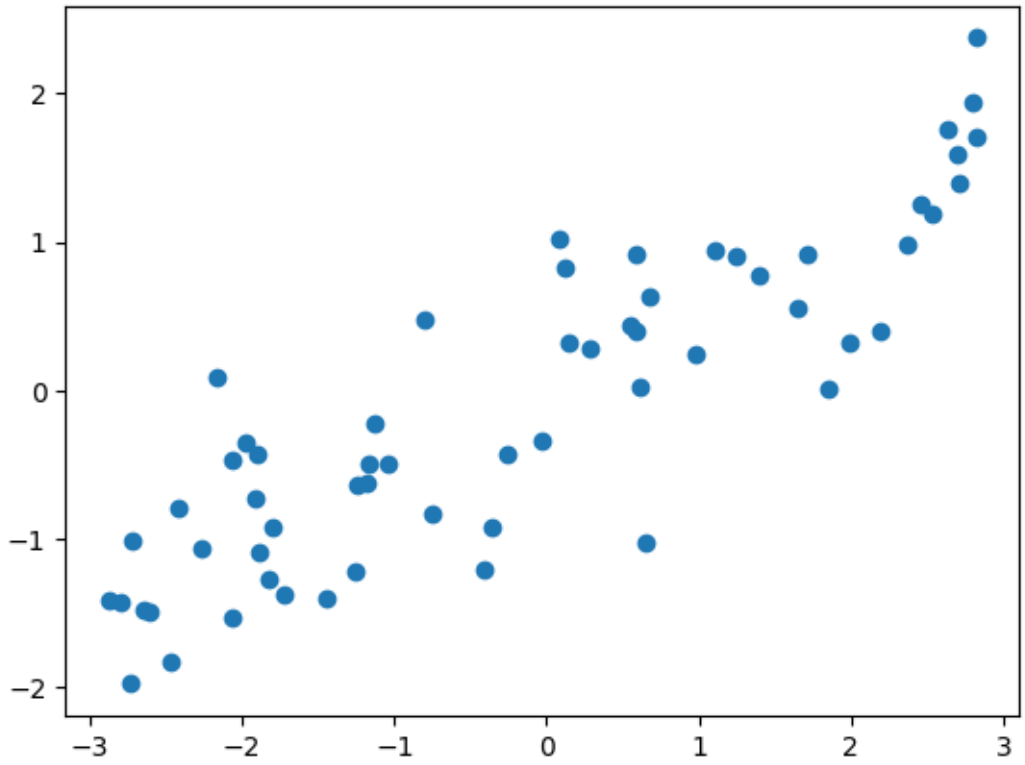
1 Least Squares

[1]:
```
import numpy as np
def make_wave(n_samples=100):
    rnd = np.random.RandomState(42)
    x = rnd.uniform(-3, 3, size=n_samples)
    y_no_noise = (np.sin(5 * x) + x)
    y = (y_no_noise + rnd.normal(size=len(x))) / 2
    return x.reshape(-1, 1), y
```

[2]:
```
import matplotlib.pyplot as plt

X, y = make_wave(n_samples=60)
plt.scatter(X, y)
```

[2]:
```
<matplotlib.collections.PathCollection at 0x7baa01307a10>
```



[3]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
lr = LinearRegression().fit(X_train, y_train)
```

[4]:
```
lr.coef_
```

[4]:
```
array([0.4966359])
```

[5]:
```
lr.intercept_
```

[5]:
```
np.float64(-0.003547055384333392)
```

[6]:
```
lr.score(X_train, y_train)
```

[6]:
```
0.7412171276539806
```

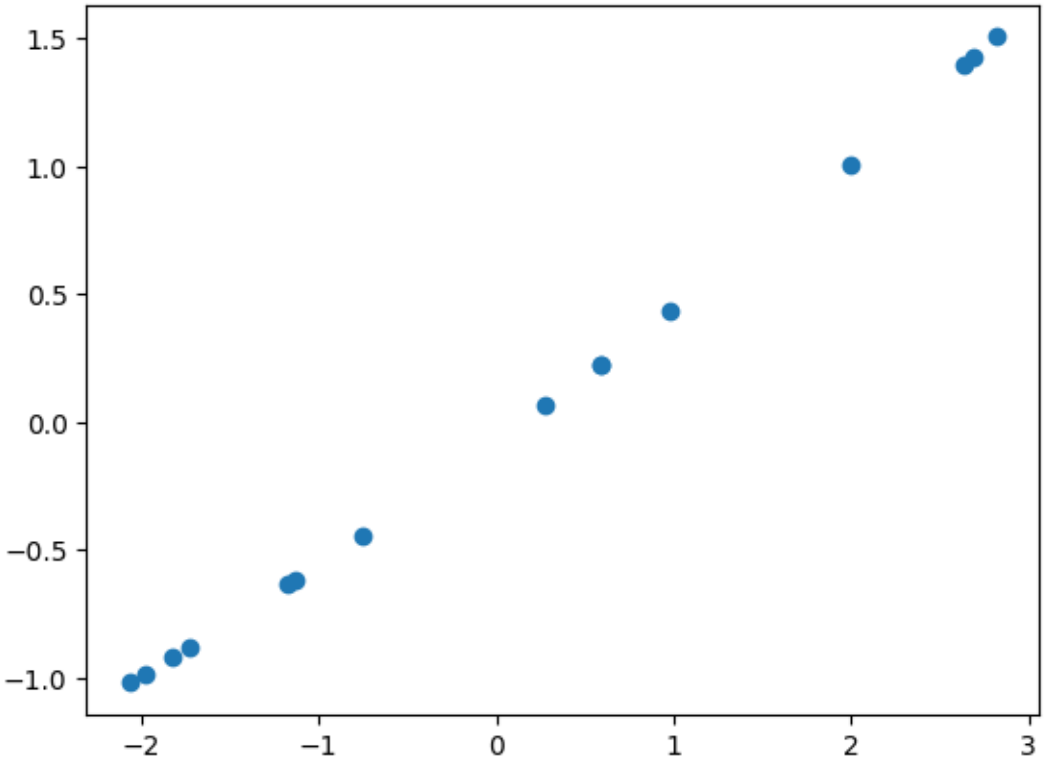
[7]:
```
lr.score(X_test, y_test)
```

[7]:
```
0.8400610110988681
```

[8]:
```
X_train_ext = np.concatenate((X_train, X_train**2), axis = 1)
X_test_ext = np.concatenate((X_test, X_test**2), axis = 1)
```

[9]:
```
lr = LinearRegression().fit(X_train_ext, y_train)
y_hat = lr.predict(X_test_ext)
%matplotlib inline
plt.scatter(X_test, y_hat)
```

[9]: <matplotlib.collections.PathCollection at 0x7ba9947747d0>



[10]:
```
print(lr.score(X_train_ext, y_train))
print(lr.score(X_test_ext, y_test))
```

```
0.7443612095410588
0.8433574307084799
```

[11]:
```
from sklearn.datasets import load_diabetes
diabetes = load_diabetes()
diabetes['data'].shape
```

[11]: (442, 10)

[12]:
```
from sklearn.preprocessing import PolynomialFeatures
X = PolynomialFeatures(degree = 2, include_bias=False).
  ↪fit_transform(diabetes['data'])
y = diabetes['target']
X.shape
```

[12]:
```
(442, 65)
```

[13]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
lr = LinearRegression().fit(X_train, y_train)
```

[14]:
```
print(lr.score(X_train, y_train))
print(lr.score(X_test, y_test))
```

```
0.604815329837055
0.42424194594593756
```

2 Ridge Regression

[15]:
```
from sklearn.linear_model import Ridge
ridge = Ridge().fit(X_train, y_train)
ridge.score(X_train, y_train)
```

[15]:
```
0.42791319284620444
```

[16]:
```
ridge.score(X_test, y_test)
```

[16]:
```
0.4387018239867466
```

[17]:
```
ridge10 = Ridge(alpha=10).fit(X_train, y_train)
ridge10.score(X_train, y_train)
```

[17]:
```
0.15099790967423454
```

[18]:
```
ridge10.score(X_test, y_test)
```

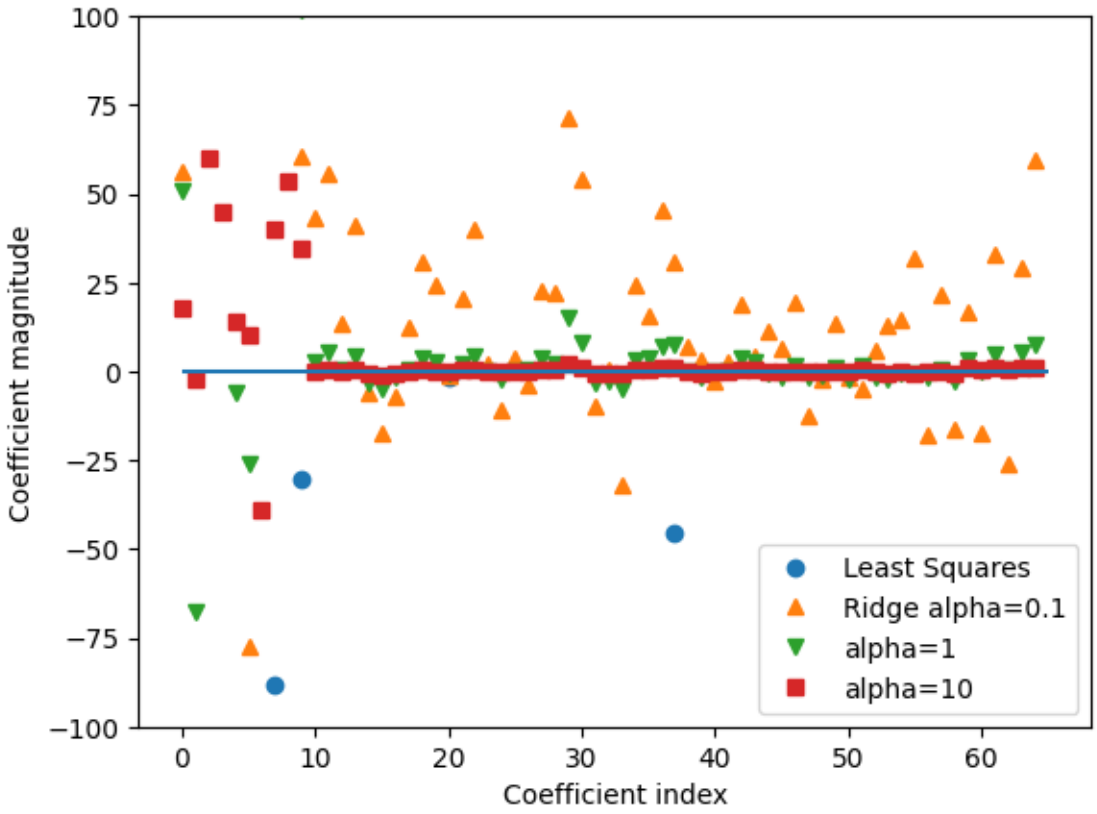
[18]:
```
0.15644986167540842
```

[19]:
```
ridge01 = Ridge(alpha=0.1).fit(X_train, y_train)

plt.plot(lr.coef_ , 'o', label="Least Squares")
plt.plot(ridge01.coef_ , '^', label="Ridge alpha=0.1")
plt.plot(ridge.coef_ ,'v',label="alpha=1")
plt.plot(ridge10.coef_ ,'s',label="alpha=10")
plt.xlabel("Coefficient index")
plt.ylabel("Coefficient magnitude")
```

```
plt.hlines(0,0,len(lr.coef_))
plt.ylim(-100,100)
plt.legend()
```

[19]: <matplotlib.legend.Legend at 0x7ba995df1410>



3 Lasso

[20]:
```
from sklearn.linear_model import Lasso
lasso = Lasso().fit(X_train, y_train)
lasso.score(X_train, y_train)
```

[20]: 0.34687336241711

[21]:
```
lasso.score(X_test, y_test)
```

[21]: 0.3791413953419158

[22]:
```
np.sum(lasso.coef_ != 0)
```

[22]:
```
np.int64(3)
```

[23]:
```
lasso001 = Lasso(alpha=0.014, max_iter = 100000).fit(X_train, y_train)
lasso001.score(X_train, y_train)
```

[23]:
```
0.5261606240788235
```

[24]:
```
lasso001.score(X_test, y_test)
```

[24]:
```
0.5016587090113085
```

[25]:
```
np.sum(lasso001.coef_ != 0)
```

[25]:
```
np.int64(13)
```

[26]:
```
lasso00001 = Lasso(alpha = 0.0001, max_iter=100000).fit(X_train, y_train)
lasso00001.score(X_train, y_train)
```

[26]:
```
0.6011668318910368
```

[27]:
```
lasso00001.score(X_test, y_test)
```

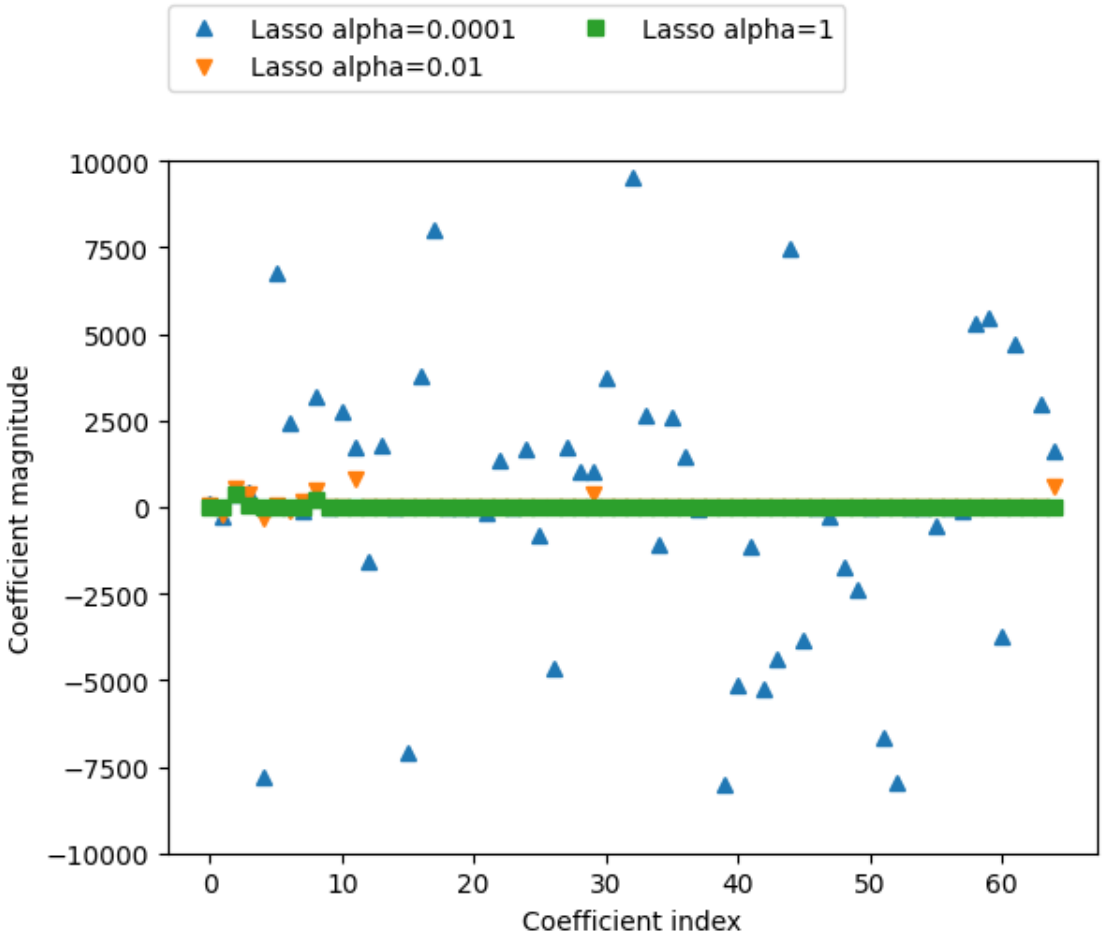
[27]:
```
0.447947925626335
```

[28]:
```
np.sum(lasso00001.coef_ != 0)
```

[28]:
```
np.int64(55)
```

[29]:
```
plt.plot(lasso00001.coef_, '^', label="Lasso alpha=0.0001")
plt.plot(lasso001.coef_, 'v', label="Lasso alpha=0.01")
plt.plot(lasso.coef_, 's', label="Lasso alpha=1")
plt.legend(ncol=2,loc=(0,1.10))
plt.ylim(-10000,10000)
plt.xlabel("Coefficient index")
plt.ylabel("Coefficient magnitude")
```

[29]:
```
Text(0, 0.5, 'Coefficient magnitude')
```



4 Exercises

1. Yes since the tigher the range Alpha is, the closer to 0 the coefficients of Ridge becomes.
2. It shows where the y-value of 0 is and shows the different coefficients in relation to it.
3. The smaller alpha becomes, the closer to zero the coefficients of lasso are. This is the opposite of what ridge regression is.
4. It is keeping a counter of all coefficients within lasso that is == to zero.
5. It indicates the # of labels that the legend can fit horizontally + the padding between the legend and the graph itself.