Lab5

November 7, 2025

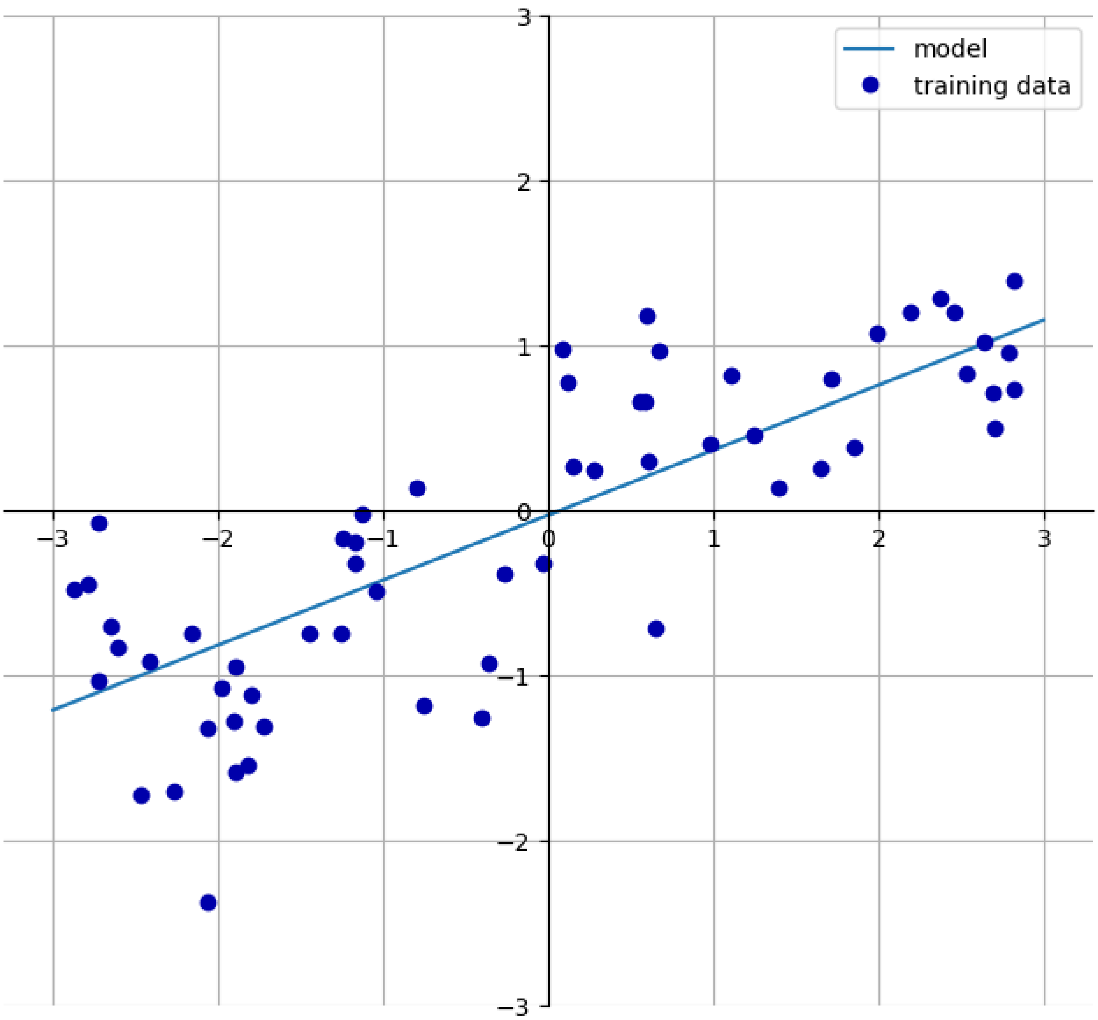
1 Lab 5

1.1 Using mglearn

[1]:
```
import mglearn
```

[2]:
```
mglearn.plots.plot_linear_regression_wave()
```

```
w[0]: 0.393906  b: -0.031804
```



[ ]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
X, y = mglearn.datasets.make_wave(n_samples=60)
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([0.39390555])
```

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

[5]:
```
np.float64(-0.031804343026759746)
```

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

[6]:
```
0.6700890315075756
```

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

[7]:
```
0.65933685968637
```

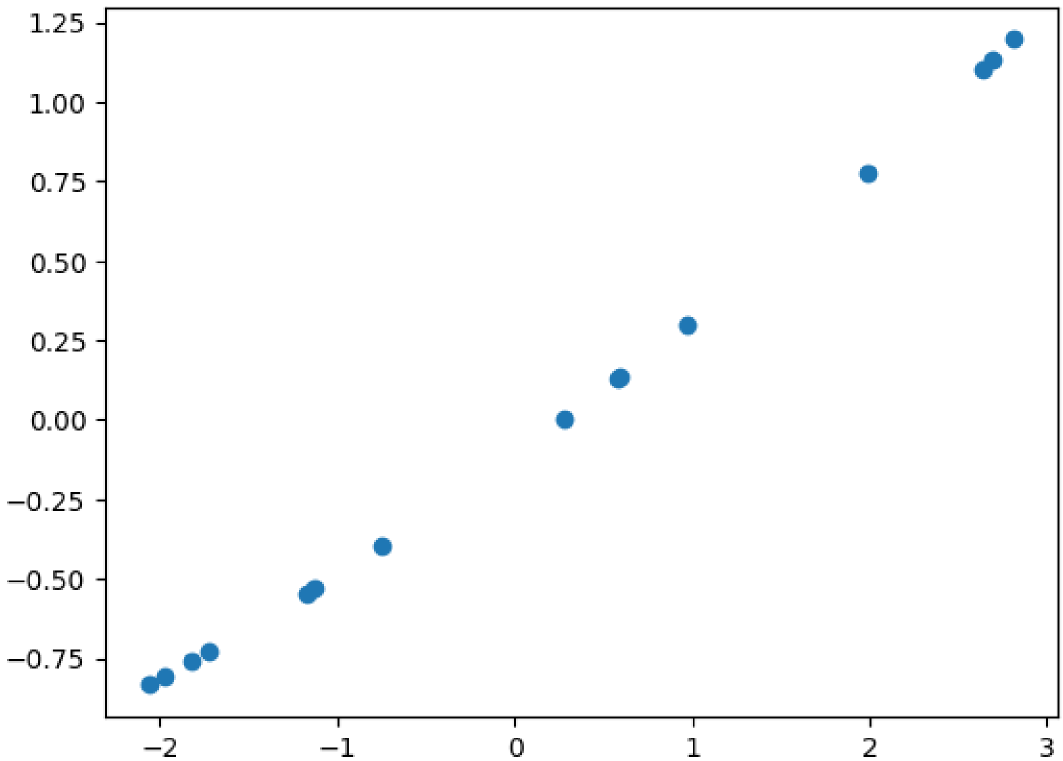
Making a feature in the wave dataset.

[8]:
```
import numpy as np
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
import matplotlib.pyplot as plt
plt.scatter(X_test, y_hat)
```

[9]:
```
<matplotlib.collections.PathCollection at 0x1a7aeac12b0>
```



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

```
0.6753386938063954
0.6377346604142683
```

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

[11]: (442, 10)

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

[12]: (442, 65)

[ ]:
```python
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))
```

```
0.6048153298370549
0.4242419459459559
```

1.2 Ridge Regression

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

[ ]: 0.42791319284620444

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

[16]: 0.43870182398674684

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

[ ]: 0.15099790967423465

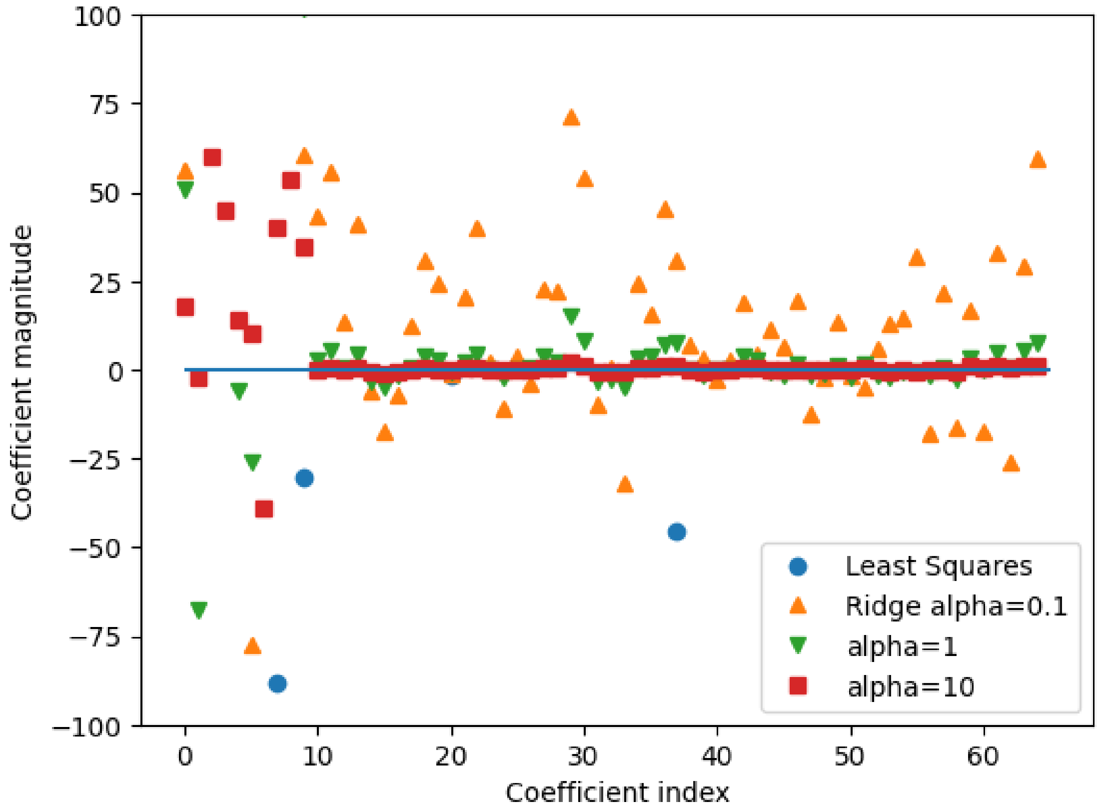
[ ]:
```
ridge10.score(X_test, y_test)
```

[ ]: 0.15644986167540842

[20]:
```
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()
```

[20]: <matplotlib.legend.Legend at 0x1a7aec8ec90>



1.3 Lasso

[21]:
```
from sklearn.linear_model import Lasso
lasso = Lasso().fit(X_train, y_train)
lasso.score(X_train, y_train)
```

[21]: 0.34687336241711

[22]:
```
lasso.score(X_test,y_test)
```

[22]: 0.3791413953419158

[23]:
```
np.sum(lasso.coef_ != 0)
```

[23]: np.int64(3)

[24]:
```
lasso001 = Lasso(alpha=0.01,max_iter=100000).fit(X_train,y_train)
lasso001.score(X_train,y_train)
```

[24]: 0.5375042588015271

[25]:
```
lasso001.score(X_test,y_test)
```

[25]: 0.5144146595591164

[26]:
```
np.sum(lasso001.coef_ != 0)
```

[26]: np.int64(16)

[27]:
```
lasso00001 = Lasso(alpha=0.0001,max_iter=100000).fit(X_train,y_train)
lasso00001.score(X_train,y_train)
```

[27]: 0.6011668318910367

[28]:
```
lasso00001.score(X_test,y_test)
```

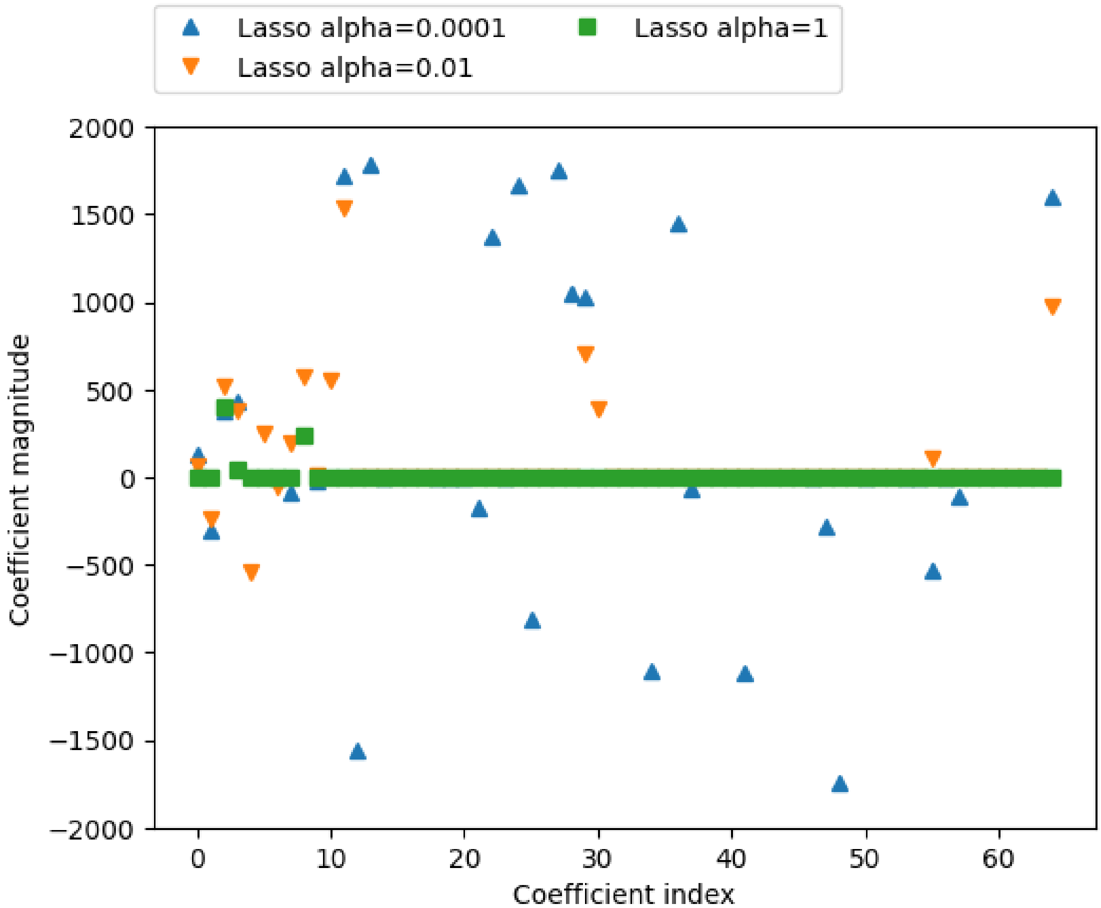
[28]: 0.4479479256263268

[29]:
```
np.sum(lasso00001.coef_ != 0)
```

[29]: np.int64(55)

[31]:
```
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=(0,1.05))
plt.ylim(-2000,2000)
plt.xlabel("Coefficient index")
plt.ylabel("Coefficient magnitude")
```

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



1.4 Exercises

1. The low value of alpha allows larger coefficient values, and a higher alpha gives a lower variation in coefficients, with a stronger bias.
2. It draws a horizontal line at $y = 0$ and splits the axis into positive and negative
3.
4. It counts the number of coefficients that are not 0.
5. The `legend` function displays the box which specificies all the different labels