Linear Regression using Normal Equation

Jisen Fang

Original code: Chapter 4 - Training Linear Models - Colab

What I did:

1. Modify the code to read data from abalone_train.csv in the local drive instead of reading random data.
2. Modify the input array to fit the size of abalone_train.csv

Read data from local

```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt

from google.colab import files
uploaded  = files.upload() # upload the data from local

import io
abalone = pd.read_csv(
   io.BytesIO(uploaded['abalone_train.csv']),
   names=["Length", "Diameter", "Height", "Whole weight", "Shucked weight",
          "Viscera weight", "Shell weight", "Age"]) # read the data into a pandas DataFrame

X1 = abalone["Length"]

X2 = np.array(X1)
X = X2.reshape(-1, 1)

y1 = abalone["Height"]
y2 = np.array(y1)
y = y2.reshape(-1, 1)
```

Modify the input array

```
X_b = np.c_[np.ones((3320, 1)), X]  # 100 to 3320 since the size of abalone_train.csv is 3320
theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)
```

All output

```python
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt

from google.colab import files
uploaded  = files.upload()

import io
abalone = pd.read_csv(
    io.BytesIO(uploaded['abalone_train.csv']),
    names=["Length", "Diameter", "Height", "Whole weight", "Shucked weight",
           "Viscera weight", "Shell weight", "Age"])

X1 = abalone["Length"]

X2 = np.array(X1)

X = X2.reshape(-1, 1)

y1 = abalone["Height"]
y2 = np.array(y1)
y = y2.reshape(-1, 1)
```

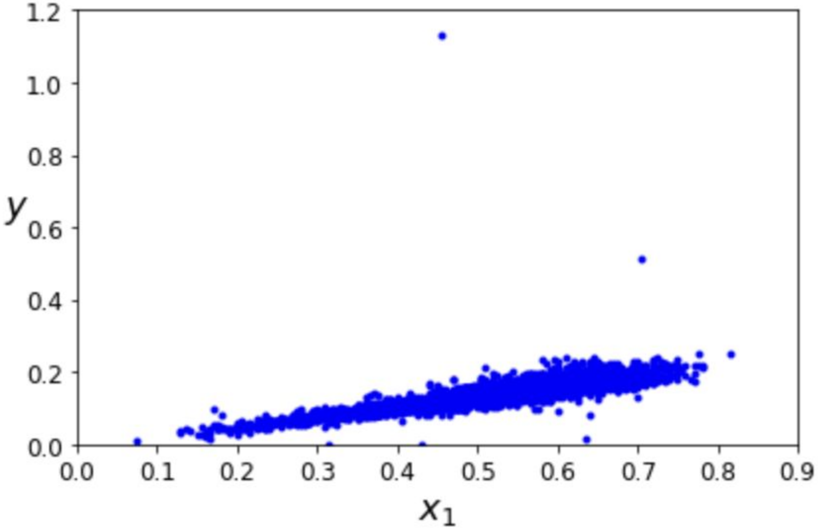
Browse... abalone_train.csv

abalone_train.csv(text/csv) - 145915 bytes, last modified: n/a - 100% done

```
Saving abalone_train.csv to abalone_train (9).csv
```

```
plt.plot(X, y, "b.")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.axis([0, 0.9, 0, 1.2])
save_fig("generated_data_plot")
plt.show()
```

```
Saving figure generated_data_plot
```



```python
X_b = np.c_[np.ones((3320, 1)), X]  # add x0 = 1 to each instance
theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)
```

[25]

```python
theta_best
```

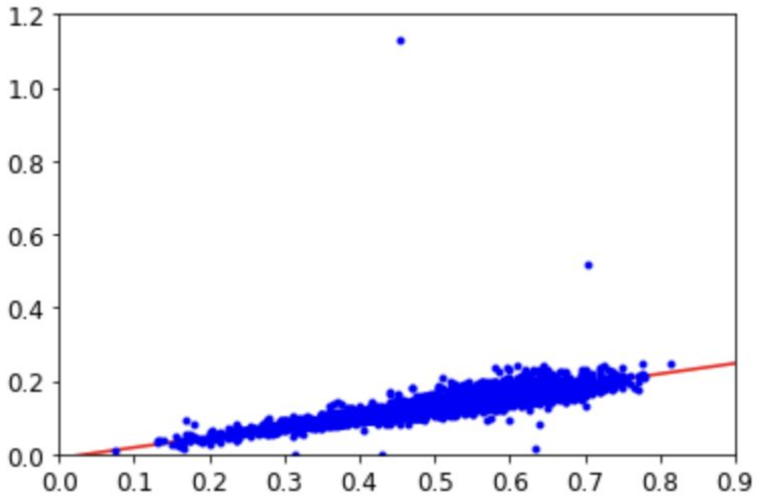
```
array([[-0.0108267 ],
       [ 0.28716253]])
```

[26]

```python
X_new = np.array([[0], [2]])
X_new_b = np.c_[np.ones((2, 1)), X_new]  # add x0 = 1 to each instance
y_predict = X_new_b.dot(theta_best)
y_predict
```

```
array([[-0.0108267 ],
       [ 0.56349837]])
```

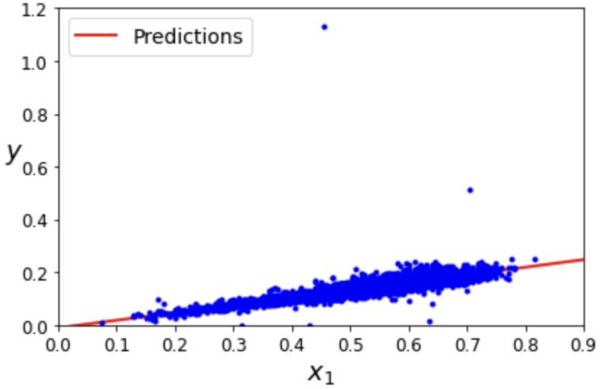
```
plt.plot(X_new, y_predict, "r-")
plt.plot(X, y, "b.")
plt.axis([0, 0.9, 0, 1.2])
plt.show()
```



[37]

```python
plt.plot(X_new, y_predict, "r-", linewidth=2, label="Predictions")
plt.plot(X, y, "b.")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.legend(loc="upper left", fontsize=14)
plt.axis([0, 0.9, 0, 1.2])
save_fig("linear_model_predictions_plot")
plt.show()
```

```
Saving figure linear_model_predictions_plot
```



[31]
```python
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(X, y)
lin_reg.intercept_, lin_reg.coef_
```

```
(array([-0.0108267]), array([[0.28716253]]))
```

[32]
```python
lin_reg.predict(X_new)
```

```
array([[-0.0108267 ],
       [ 0.56349837]])
```

[33]
```python
theta_best_svd, residuals, rank, s = np.linalg.lstsq(X_b, y, rcond=1e-6)
theta_best_svd
```

```
array([[-0.0108267 ],
       [ 0.28716253]])
```

[34]
```python
np.linalg.pinv(X_b).dot(y)
```

```
array([[-0.0108267 ],
       [ 0.28716253]])
```