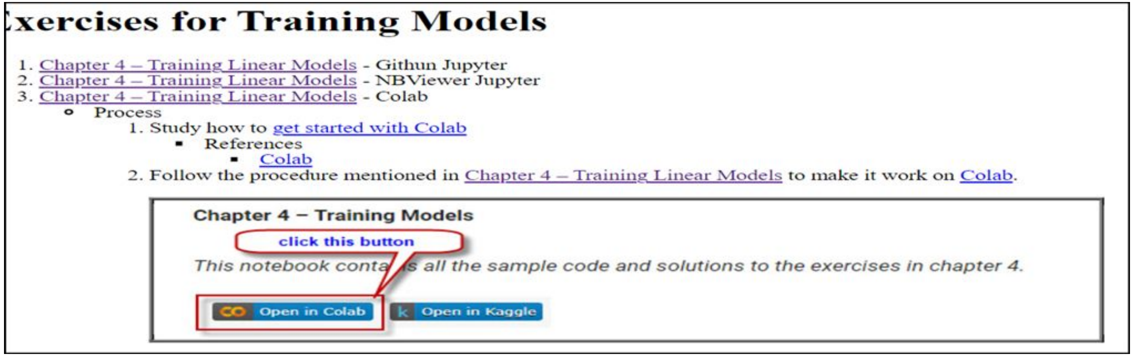
Machine Learning - Supervised Learning
Linear Regression using Normal Equation

- ❖ Supervised learning - Linear regression using Normal Equation



Exercises for Training Models

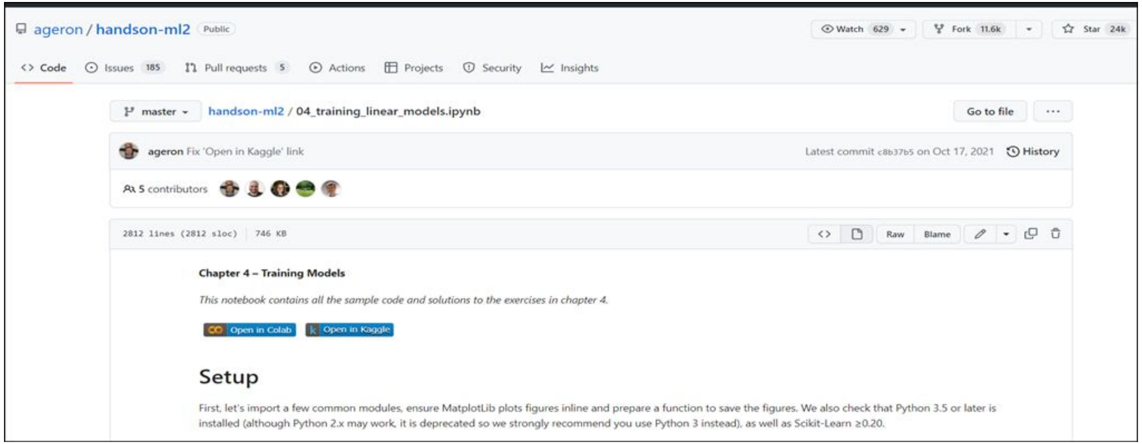
1. Chapter 4 – Training Linear Models - Githun Jupyter
2. Chapter 4 – Training Linear Models - NBViewer Jupyter
3. Chapter 4 – Training Linear Models - Colab
    - Process
        1. Study how to get started with Colab
            - References
                - Colab
        2. Follow the procedure mentioned in Chapter 4 – Training Linear Models to make it work on Colab.

Chapter 4 – Training Models

click this button

This notebook contains all the sample code and solutions to the exercises in chapter 4.

Open in Colab | Open in Kaggle



ageron / handson-ml2 Public

Watch 629 | Fork 11.6k | Star 24k

Code | Issues 185 | Pull requests 5 | Actions | Projects | Security | Insights

master | handson-ml2 / 04_training_linear_models.ipynb | Go to file

ageron Fix 'Open in Kaggle' link | Latest commit c8b37b5 on Oct 17, 2021 | History

5 contributors

2812 lines (2812 sloc) | 746 KB | Raw | Blame

Chapter 4 – Training Models

This notebook contains all the sample code and solutions to the exercises in chapter 4.

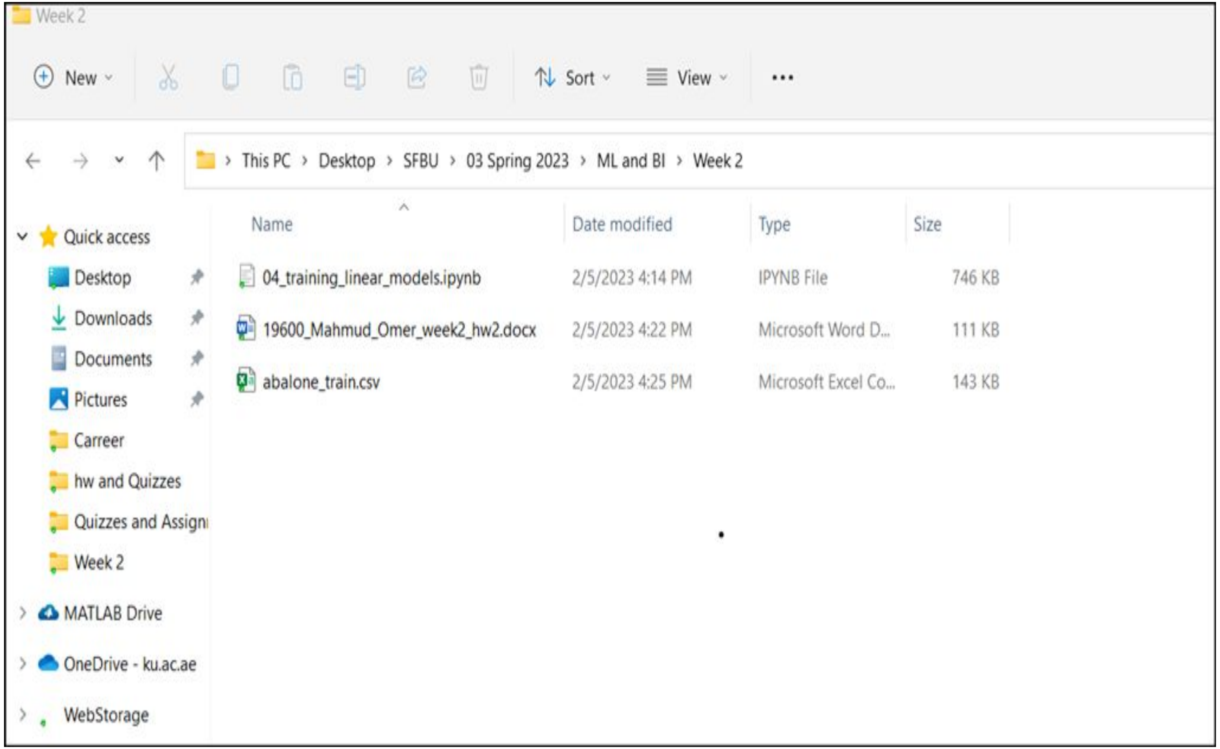
Open in Colab | Open in Kaggle

Setup

First, let's import a few common modules, ensure MatplotLib plots figures inline and prepare a function to save the figures. We also check that Python 3.5 or later is installed (although Python 2.x may work, it is deprecated so we strongly recommend you use Python 3 instead), as well as Scikit-Learn ≥0.20.

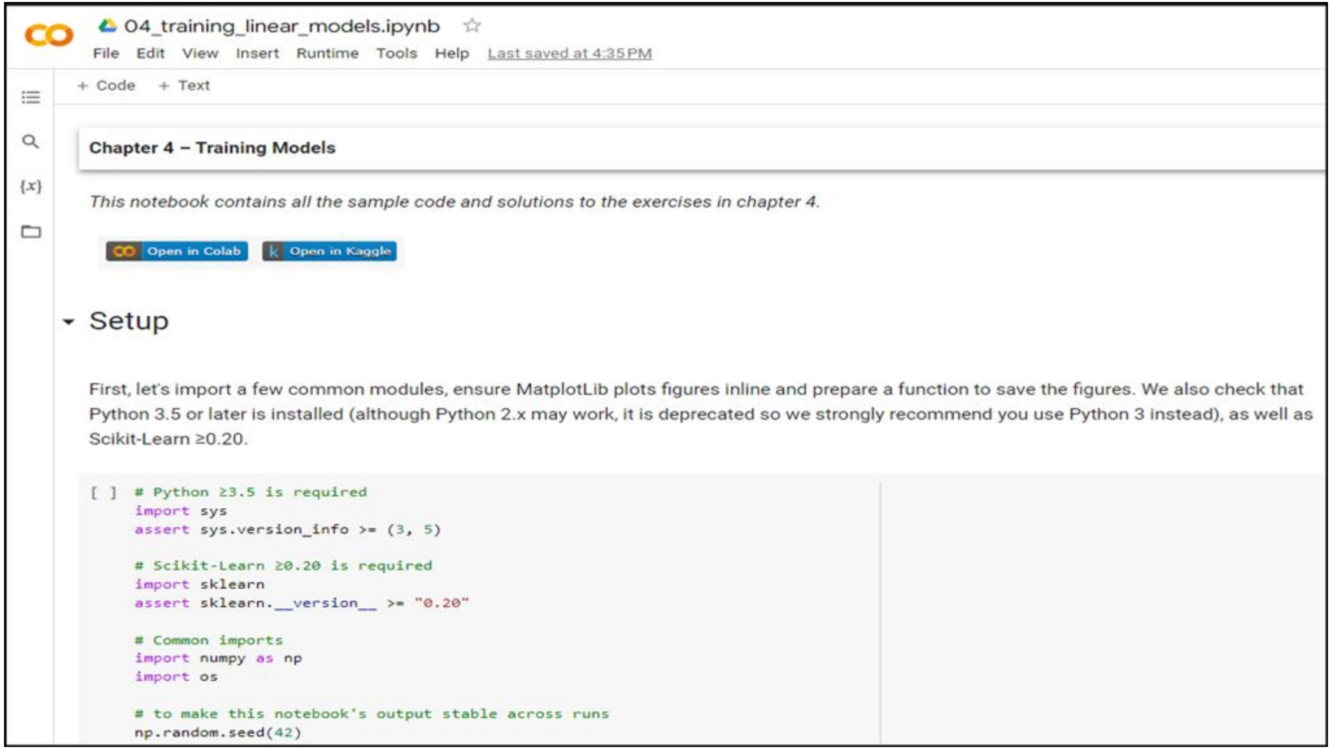
Machine Learning - Supervised Learning
Linear Regression using Normal Equation

❖ Importing Jupyter notebook file from local disc to google Colab



Machine Learning - Supervised Learning
Linear Regression using Normal Equation

- Setting up : Importing common modules, ensure MatplotLib plots figures inline and prepare a function to save the figures



Machine Learning - Supervised Learning
Linear Regression using Normal Equation

- Modifying original code in linear regression using Normal equation:

4. Change the process mentioned in Step 1 by reading CVS test data from a local drive : abalone_train.cvs
   - Process
     a. You can modify the code in Linear regression using the Normal Equation (local copy: pdf, html) - Demo

Instead of reading random data

```
import numpy as np
X = 2 * np.random.rand(100, 1)
y = 4 + 3 * X + np.random.randn(100, 1)
```

The Normal Equation

Modify these 2 lines

```
[ ] import numpy as np

    X = 2 * np.random.rand(100, 1)
    y = 4 + 3 * X + np.random.randn(100, 1)
```

```
plt.plot(X, y, "b.")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.axis([0, 2, 0, 15])
save_fig("generated_data_plot")
plt.show()
```

Machine Learning - Supervised Learning

Linear Regression using Normal Equation

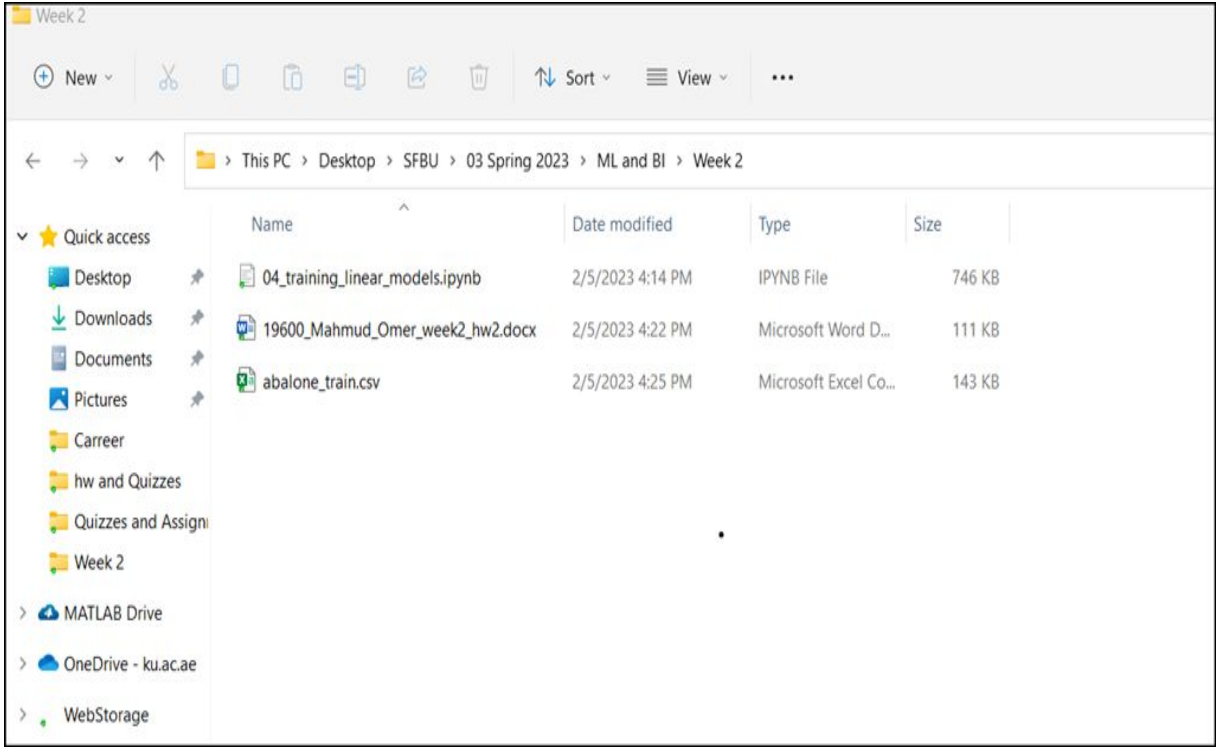
- uploading files from local disc for modified code

```
import numpy as np
import pandas as pd

# X = 2 * np.random.rand(100, 1)
# y = 4 + 3 * X + np.random.randn(100, 1)
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 is
#     0        0.435
#     1        0.585
#     2        0.655
#     .....
X1 = abalone["Length"]

# X2 is
#     array([0.435, 0.585, ...., 0.45])
X2 = np.array(X1)

# X is
#     array([[0.435],
#            [0.585],
#            [0.655],
#            ...,
#            [0.53 ],
#            [0.395],
#            [0.45 ]])
X = X2.reshape(-1, 1)

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

Choose Files No file chosen Cancel upload

Machine Learning - Supervised Learning
Linear Regression using Normal Equation

- ❖ uploading abalone_train.csv file from local disc for modified code



Machine Learning - Supervised Learning
Linear Regression using Normal Equation

❖ Uploaded csv file , and updated code result

```
Choose Files abalone_train.csv
• abalone_train.csv(text/csv) - 145915 bytes, last modified: 2/5/2023 - 100% done
Saving abalone_train.csv to abalone_train.csv
```

```
[7] 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)
```

Machine Learning - Supervised Learning

Linear Regression using Normal Equation

- Linear regression for uploaded csv file data , and updated code output result

```
[7] 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)
```

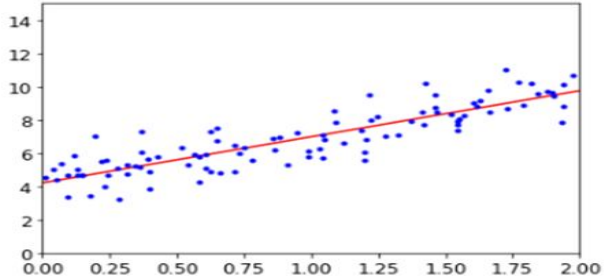
```
[ ] theta_best
```

```
array([[4.21509616],
       [2.77011339]])
```

```
[ ] 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([[4.21509616],
       [9.75532293]])
```

```
plt.plot(X_new, y_predict, "r-")
plt.plot(X, y, "b.")
plt.axis([0, 2, 0, 15])
plt.show()
```



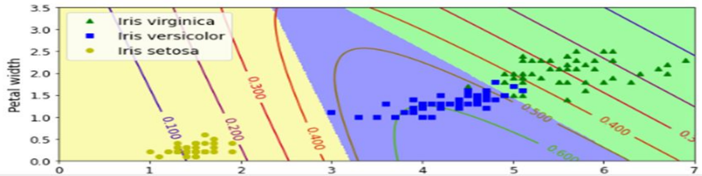
Machine Learning - Supervised Learning

Linear Regression using Normal Equation

- All errors has been fixed , the code was updated and every single line of code run successfully

```
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"],nrows = 100)
```

```
[93] x0, x1 = np.meshgrid(
             np.linspace(0, 8, 500).reshape(-1, 1),
             np.linspace(0, 3.5, 200).reshape(-1, 1),
         )
     X_new = np.c_[x0.ravel(), x1.ravel()]
     X_new_with_bias = np.c_[np.ones([len(X_new), 1]), X_new]

     logits = X_new_with_bias.dot(Theta)
     Y_proba = softmax(logits)
     y_predict = np.argmax(Y_proba, axis=1)

     zz1 = Y_proba[:, 1].reshape(x0.shape)
     zz = y_predict.reshape(x0.shape)

     plt.figure(figsize=(10, 4))
     plt.plot(X[y==2, 0], X[y==2, 1], "g^", label="Iris virginica")
     plt.plot(X[y==1, 0], X[y==1, 1], "bs", label="Iris versicolor")
     plt.plot(X[y==0, 0], X[y==0, 1], "yo", label="Iris setosa")

     from matplotlib.colors import ListedColormap
     custom_cmap = ListedColormap(['#fafab0','#9898ff','#a0faa0'])

     plt.contourf(x0, x1, zz, cmap=custom_cmap)
     contour = plt.contour(x0, x1, zz1, cmap=plt.cm.brg)
     plt.clabel(contour, inline=1, fontsize=12)
     plt.xlabel("Petal length", fontsize=14)
     plt.ylabel("Petal width", fontsize=14)
     plt.legend(loc="upper left", fontsize=14)
     plt.axis([0, 7, 0, 3.5])
     plt.show()
```



Machine Learning - Supervised Learning

Linear Regression using Normal Equation

```
[94] logits = X_test.dot(Theta)
     Y_proba = softmax(logits)
     y_predict = np.argmax(Y_proba, axis=1)

     accuracy_score = np.mean(y_predict == y_test)
     accuracy_score

     0.9333333333333333
```

Our perfect model turns out to have slight imperfections. This variability is likely due to the very small size of the dataset: depending on how you sample the training set, validation set and the test set, you can get quite different results. Try changing the random seed and running the code again a few times, you will see that the results will vary.

```
[94]
```

Google slides and GitHub Links

https://docs.google.com/presentation/d/1CAznmiCZUZ8I4RJ0WW1k1NIIWfW5ahN8WRBblzhybp0/edit?usp=sharing

https://github.com/momer22/Machine-Learning---Supervised-Learning---Linear-Regression-using-Normal-Equation