Python Data Products

Course 2: Design thinking and predictive pipelines

Lecture: Regression in Python

Learning objectives

In this lecture we will...

- Explore how to express linear regression equations in terms of Python data structures
- Work through a (simple) **real-world regression example**
- Compare a "manual" implementation of linear regression to a library function

Example – Air quality prediction

We'll look at the problem of predicting **air quality,** using an index called pm2.5, measured in Beijing

- This is a "simpler" dataset than some of the others we've been working with, as the relevant features are all real-valued
- It's also useful in our following lecture (on time-series prediction), since the data is in the form of a time series

Example – UCI Dataset Repository

Beijing PM2.5 Data Data Set

Download: Data Folder, Data Set Description

Abstract: This hourly data set contains the PM2.5 data of US Embassy in Beijing. Meanwhile, meteorological data from Beijing Capital International Airport are also included.

Data Set Characteristics:	Multivariate, Time-Series	**Number of Instances:**	43824	**Area:**	Physical
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	13	**Date Donated**	2017-01-19
Associated Tasks:	Regression	**Missing Values?**	Yes	**Number of Web Hits:**	70637

Source:

Song Xi Chen, csx '@' gsm.pku.edu.cn, Guanghua School of Management, Center for Statistical Science, Peking University.

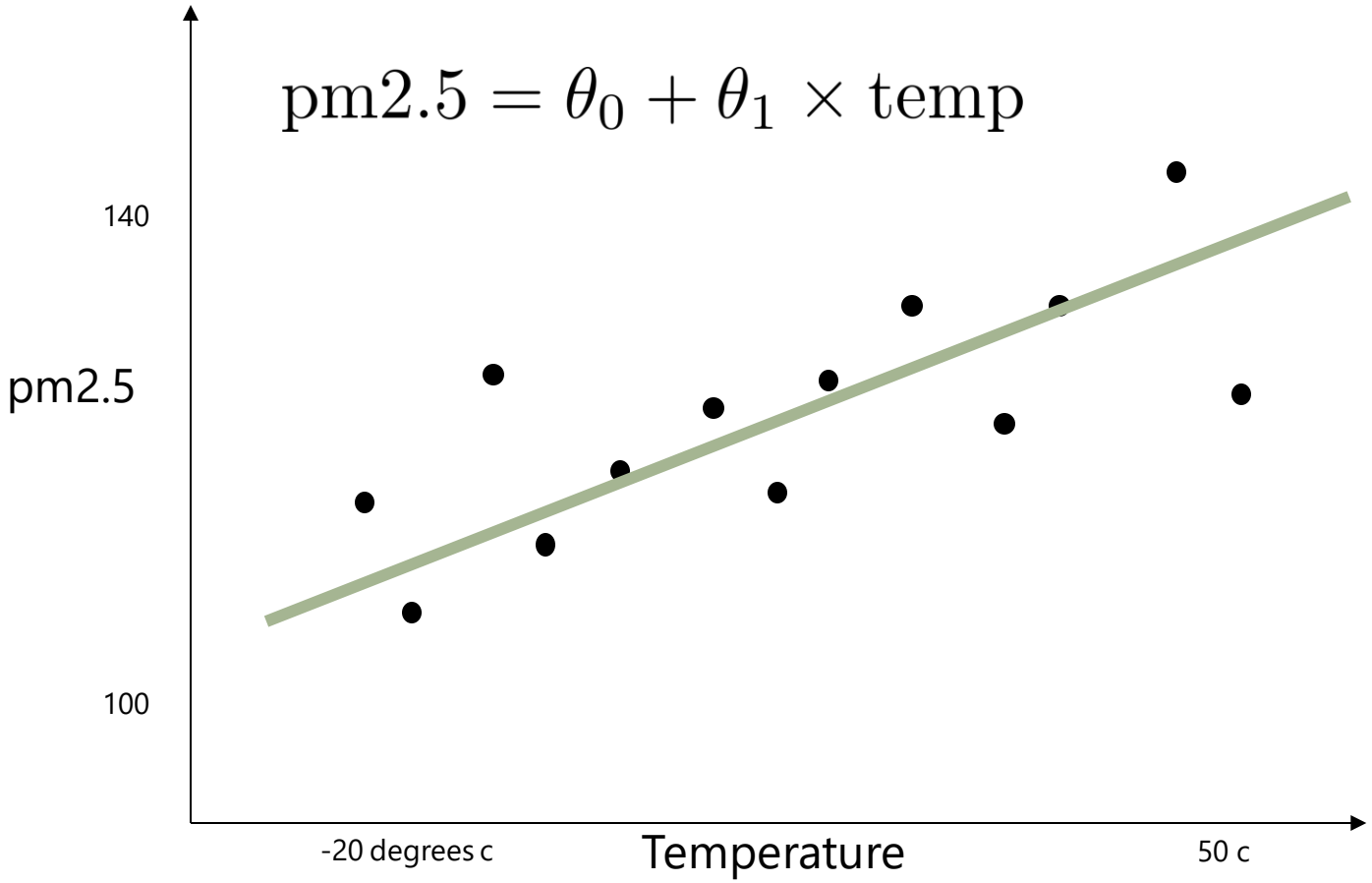
Data Set Information:

The dataâ€™s time period is between Jan 1st, 2010 to Dec 31st, 2014. Missing data are denoted as â€œNAâ€.

https://archive.ics.uci.edu/ml/datasets/Beijing+PM2.5+Data

What are we trying to predict?

E.g. pm2.5 vs. Temperature:



Code: Reading the file

```
In [2]: path = "datasets/PRSA_data_2010.1.1-2014.12.31.csv"
        f = open(path, 'r')
```

```
In [3]: dataset = []
        header = f.readline().strip().split(',')
        for line in f:
            line = line.split(',')
            dataset.append(line)
```

```
In [4]: N = len(dataset)
        N
```

```
Out[4]: 43824
```

```
In [5]: header
```

```
Out[5]: ['No',
         'year',
         'month',
         'day',
         'hour',
         'pm2.5',
         'DEWP',
         'TEMP',
         'PRES',
         'cbwd',
         'Iws',
```

Label that we want to predict

Feature we want to use for prediction

Code: Extracting features and labels

```
In [6]: header.index('pm2.5')
Out[6]: 5

In [7]: header.index('TEMP')
Out[7]: 7

In [8]: y = [float(d[5]) for d in dataset]
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-8-caefd9ca5537> in <module>()
----> 1 y = [float(d[5]) for d in dataset]

<ipython-input-8-caefd9ca5537> in <listcomp>(.0)
----> 1 y = [float(d[5]) for d in dataset]

ValueError: could not convert string to float: 'NA'

In [9]: dataset = [d for d in dataset if d[5] != 'NA']
```

Code: Let's try again…

```
In [10]: y = [float(d[5]) for d in dataset]
```

```
In [11]: def feature(datum):
             feat = [1, float(datum[7])]
             return feat
```

```
In [12]: X = [feature(d) for d in dataset]
```

```
In [13]: y[:10]
```

```
Out[13]: [129.0, 148.0, 159.0, 181.0, 138.0, 109.0, 105.0, 124.0, 120.0, 132.0]
```

```
In [14]: X[:10]
```

```
Out[14]: [[1, -4.0],
          [1, -4.0],
          [1, -5.0],
          [1, -5.0],
          [1, -5.0],
          [1, -6.0],
          [1, -6.0],
          [1, -5.0],
          [1, -6.0],
          [1, -5.0]]
```

Reminder: Constant feature

Why did we implement our feature function like this?

In [11]:
```
def feature(datum):
    feat = [1, float(datum[7])]
    return feat
```

Code: Finding the parameters

```
In [15]: theta,residuals,rank,s = numpy.linalg.lstsq(X, y)

In [16]: theta
Out[16]: array([107.10183392,  -0.68447989])
```

pm2.5 = 107.1 - 0.68 * temp

Code: Adding more features

```
In [17]: def feature(datum):
             feat = [1, float(datum[7]), float(datum[8]), float(datum[10])]
             return feat

In [18]: X = [feature(d) for d in dataset]

In [19]: theta,residuals,rank,s = numpy.linalg.lstsq(X, y)

In [20]: theta
Out[20]: array([ 3.26373064e+03, -3.10933772e+00, -3.06517728e+00, -4.60017221e-01])
```

Note: pressure

Note: wind speed

pm2.5 = 3263.7
- 3.109 * temp
- 3.065 * pressure
- 0.460 * wind speed

Code: Doing the same thing manually

```
In [20]: theta
Out[20]: array([ 3.26373064e+03, -3.10933772e+00, -3.06517728e+00, -4.60017221e-01])

In [21]: X = numpy.matrix(X)
         y = numpy.matrix(y)
         numpy.linalg.inv(X.T * X) * X.T * y.T
Out[21]: matrix([[ 3.26373064e+03],
                 [-3.10933772e+00],
                 [-3.06517728e+00],
                 [-4.60017221e-01]])
```

Summary of concepts

- Demonstrated how to perform simple linear regression in Python
- Performed linear regression on an "air quality" example from the UCI Machine Learning Repository
- Introduced the numpy "least squares" function for linear regression

On your own...

- Try extending the code provided here to use different features and feature combinations