20211020 Lecture notes

Wednesday, October 20, 2021 4:31 PM

- Midterm
  - What is a good score?
    - Bare minimum: Need to do better than the code he gave us
    - Less than < 1 pretty OK
    - less than < 0.7 excellent
  - What packages can you use?
  - You creep up the leader board by discovering features
    - What words are unique to, are used in X and not score Y, percentage of different words are using
      - TFDIF
        - May not even run on your computer
  - Taking a representative sample of the dataset is not easy to do
  - Focus on a model that makes sense, be able to explain your process and workflow
    - Feature engineering techniques
    - How did you decide what features to include
    - What models did you try
    - How did you evaluate how well the model did
    - Your process
  - If you don't know if it's deep learning, don't use it
    - Look up the definition of deep learning…

20211020 Linear Regression

Wednesday, October 20, 2021 3:45 PM

-

Linear Regression

Boston University CS 506 - Lance Galletti

Challenge for those who have LR experience

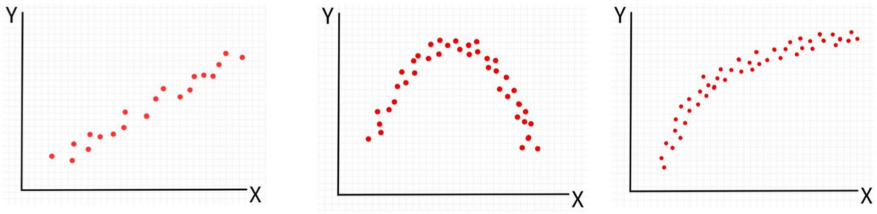
- Find the data.csv file in the regression folder of our course repo
- Challenge:
  - Every day my alarm goes off at seemingly random times...
  - I've recorded the times for the past year of so (1 - 355 days)
  - Today is day 356
  - Can you predict when my alarm will ring?

- If you have done linear regression before, than try the challenge
  - This dataset is on the repo

- It will ring during class time FOR SURE
- Try to run your linear regression model, you will have to do some feature tuning and feature extraction
- Raise hand if you have a prediction

Motivation

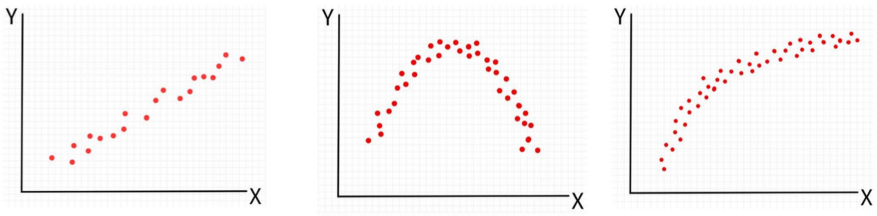
Given **n** samples / data points ($\mathbf{y_i}$, $\mathbf{x_i}$)



- We are moving away from classification
- We are now wanting to predict a continuum of possible values
- X and Y are continuous
- How does Y change as a function of X
- Any new point X, we apply learned function h and get an estimate for Y

Motivation

Understand/explain how **y** varies as a function of **x** (i.e. find a function **y = h(x)** that best fits our data)



Motivation

Suppose we are given a curve **y = h(x)**, how can we evaluate whether it is a good fit to our data?

Compare $\mathbf{h(x_i)}$ to $\mathbf{y_i}$ for all **i**.

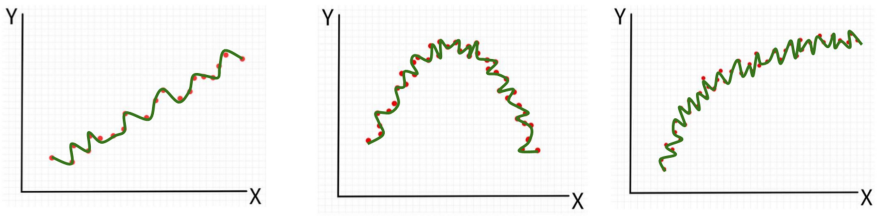
Goal: For a given distance function **d**, find **h** where **L** is smallest.

$$L(h) = \sum_i d(h(x_i), y_i)$$

- If I gave you a function h, how would you know it's a good prediction or not?
- For a particular distance function, can take the sum of all the comparisons

Motivation

Should **h** be the curve that goes through the most samples? I.e. do we want $\mathbf{h(x_i) = y_i}$ for the maximum number of **i**?

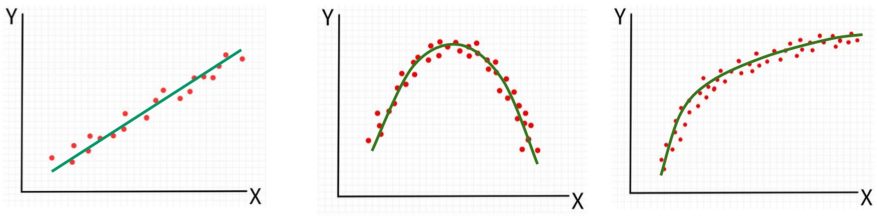


h may be too complex
overfitting - may not perform well on unseen data

- Do we want the curve to go through every point? No
  - You may overfit
  - Also over-complicated
  - We want to be able to discover h ourselves without an oracle giving us the equation
- Our cost function still stands
- It's not about finding the right distance function
- It's about constraining the functions that we want to look at
- Infinity of infinity of functions
  - Iterate through that many functions set us up for failure
- Create some constraints for what would be good candidates for us

Motivation

The following curves seem the most intuitive "best fit" to our samples. How can we define this best fit mathematically? Is it just about finding the right distance function?



Motivation

Another way to define this problem is in terms of probability.

Define **P(Y | h)** as the probability of observing **Y** given that it was sampled from **h**.

Goal: Find **h** that maximizes the probability of having observed our data.

- We just want to find the distribution h that maximizes the probability of having observed Y

Motivation

To sum up we can either:

1. Minimize

$$L(h) = \sum_i d(h(x_i), y_i)$$

1. Maximize

L(h) = P(Y | h)

- Need to parameterize h in both cases
- We need a smaller window into the space of both functions

Getting Started

Do we have enough to get started?

Seems like there are too many possible **h** and our problem statements are still too vague to effectively find solutions.

What can we do to constrain the problem?

Let's make some assumptions!

- We make some assumptions

Assumptions

Let's start by assuming our data was generated by a **linear function** plus some **noise**:

$$\vec{y} = h_\beta(\mathrm{X}) + \vec{\epsilon}$$

Where **h** is linear in a parameter $\boldsymbol{\beta}$.
Which functions below are linear in $\boldsymbol{\beta}$?

$h(x) = \beta_1 x$	✔
$h(x) = \beta_0 + \beta_1 x$	✔
$h(x) = \beta_0 + \beta_1 x + \beta_2 x^2$	✔
$h(x) = \beta_1 \log(x) + \beta_2 x^2$	✔
$h(x) = \beta_0 + \beta_1 x + \beta_1^2 x$	✗

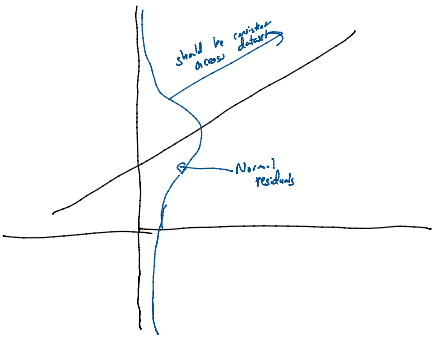
- In linear regression, we assume the data is LINEAR
- What does it mean to be linear in parameter beta
- We don't care about x, we just want the beta terms to be linear
- Green checks are linear in Beta
  - Does not need to be linear in x
- Red x are not linear in Beta
- Finding H is the same as finding all the Beta coefficients

Assumptions

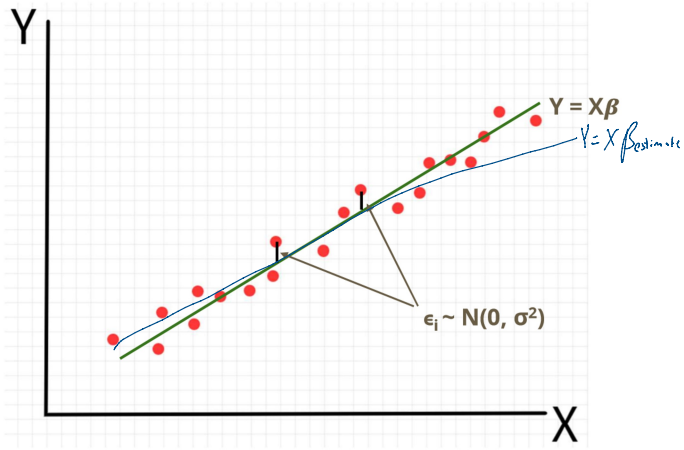
1. The relation between **x** (independent variable) and **y** (dependent variable) is linear in a parameter $\boldsymbol{\beta}$.
2. ϵ_i are independent, identically distributed random variables following a $\mathbf{N(0, \sigma^2)}$ distribution. (Note: $\boldsymbol{\sigma}$ is constant)

-

-



Assumptions



- Y = XB generated the data
- There will always be noise
- We might never observe a point on the line
- We would like to get an estimate for Y = XB
- There should be an updated version of this slide
- We want on average to fall on the line

Goal

Given these assumptions, let's try to solve the max and min problems we defined earlier!

Q: What does solving these mean?

A: Finding $\boldsymbol{\beta}$ is equivalent to finding **h**

Least Squares

$$\begin{aligned}\beta_{LS} &= \arg\min_{\beta} \sum_i d(h_\beta(x_i), y_i) \\ &= \arg\min_{\beta} \|\vec{y} - h_\beta(\mathrm{X})\|_2^2 \\ &= \arg\min_{\beta} \|\vec{y} - \beta\mathrm{X}\|_2^2\end{aligned}$$

- Let's choose Euclidean distance and square it
- This is matrix notation

Least Squares

$$\begin{aligned}\frac{\partial}{\partial\beta} &= 0 \\ \frac{\partial}{\partial\beta}(y-\beta\mathrm{X})^T(y-\beta\mathrm{X}) &= 0 \\ \frac{\partial}{\partial\beta}(y^Ty - y^TX\beta - \beta^TX^Ty - \beta^TX^TX\beta) &= 0 \\ \frac{\partial}{\partial\beta}(y^Ty - 2\beta^TX^Ty - \beta^TX^TX\beta) &= 0 \\ -2X^Ty - X^TX\beta &= 0 \\ X^TX\beta &= X^Ty \\ \boxed{\beta_{LS} = (X^TX)^{-1}X^Ty}\end{aligned}$$

- Matrix version
  - Needed for more that than two variables
  - We don't need to get too deep into this math in this class
- It's like magic!
  - Squared equation gets you an estimate of Beta

- Beta least square parameter that you can get directly from your data

Maximum Likelihood

Since $\epsilon \sim N(0, \sigma^2)$ and $Y = X\beta + \epsilon$ then $Y \sim N(X\beta, \sigma^2)$.

$$\begin{aligned}
\beta_{MLE} &= \arg\max_{\boldsymbol{\beta}} \frac{1}{\sqrt{(2\pi)^n \sigma^n}} \exp\left(-\frac{\|y - X\beta\|_2^2}{2\sigma^2}\right) \\
&= \arg\max_{\boldsymbol{\beta}} \exp\left(-\|y - X\beta\|_2^2\right) \\
&= \arg\max_{\boldsymbol{\beta}} -\|y - X\beta\|_2^2 \\
&= \arg\min_{\boldsymbol{\beta}} \|y - X\beta\|_2^2 \\
&= \beta_{LS} = (X^T X)^{-1} X^T y
\end{aligned}$$

- This is the other approach
- Maximum Likelihood Estimation
- We know how Y is distributed
  - We want to find h that maximizes this probability
- Evaluate the PDF of the normal distribution
- Maximizing the exponent
- Now we got the same equation
  - We had two approaches to this problem, converged on the same solution
- Get same estimate for Beta using this approach as well

An Unbiased Estimator

$\boldsymbol{\beta}_{LS}$ is an unbiased estimator of the true $\boldsymbol{\beta}$. That is $\mathbf{E}[\boldsymbol{\beta}_{LS}]=\boldsymbol{\beta}$.

$$
\begin{aligned}
E[\beta_{LS}] &= E[(X^T X)^{-1} X^T y] \\
&= (X^T X)^{-1} X^T E[y] \\
&= (X^T X)^{-1} X^T E[X\beta + \epsilon] \\
&= (X^T X)^{-1} X^T X\beta + E[\epsilon] \\
&= \beta
\end{aligned}
$$

sub

$\rightarrow 0$

- Beta least squares is an unbiased estimator
  - On average, you get the true Beta
- How do we show this is the case?
  - Derivation
- Guarantee provided you are following the assumptions
  - If your data is indeed generated from this function…
- Expectation of beta least squares
- Random variable
  - y is random, X is not random
- Expectation of epsilon is 0

Demo

Logistic Regression

So far y_i was a continuous variable. What if y_i is categorical?

Assume we have **2 classes**.

Even if we can make these classes numerical (i.e. translate labels such as "yes"/"no" into 1 / 0), these numbers don't have a mathematical meaning in the context of linear models and what we learn will be as arbitrary as the numerical labels we assigned (i.e. using "yes" =2/"no"=7 instead of "yes"=1/"no"=0 might "fit" a better model...).

Maybe we can use the probability of belonging to a given class as a proxy for how confidently we can classify a given point? Maybe we can fit a linear model to the probability of being in a given class!

Logistic Regression

So the output of our regression model could be a probability. But how can we enforce that $X\boldsymbol{\beta}_{LS}$ from our model is always constrained to [0,1]? i.e. how can we learn a $\boldsymbol{\beta}_{LS}$ such that $0 \leq X\boldsymbol{\beta}_{LS} \leq 1$ even for unseen X?

Instead define the odds = p / 1 - p where p = P(Y = class 1 | X)

Now the range of $X\boldsymbol{\beta}_{LS}$ is [0, ∞)

But again how can we enforce that the $X\boldsymbol{\beta}_{LS}$ are constrained to [0, ∞)? We need (-∞, ∞) - but how?

Let's take the log! This is also convenient numerically because in the previous odds format, tiny variations in p have large effects on the odds!

Logistic Regression

Our goal is to fit a linear model to the log-odds of being in one of our classes (in the 2-class case) i.e.

$$\log\left(\frac{P(Y=1|X)}{1-P(Y=1|X)}\right) = \alpha + \beta X$$

Logistic Regression

Suppose we have such a model. How do we recover the P(Y=1 |X)?

$$\log(\frac{P(Y=1|X)}{1-P(Y=1|X)}) = \alpha + \beta X$$

$$\frac{P(Y=1|X)}{1-P(Y=1|X)} = e^{\alpha+\beta X}$$

$$\frac{P(Y=1|X)}{1-P(Y=1|X)} + 1 = e^{\alpha+\beta X} + 1$$

$$\frac{P(Y=1|X)}{1-P(Y=1|X)} = e^{\alpha+\beta X} + 1$$

$$P(Y=1|X) = \frac{e^{\alpha+\beta X}}{1+e^{\alpha+\beta X}}$$

The function we apply to our probability to obtain the log odds is called the **logit** function. The function used to retrieve our probability from the log odds is called **logit^{-1}**

Logistic Regression

How do we learn our model? I.e. the α and $\boldsymbol{\beta}$ parameters.

We know:

$$P(y_i = 1|x_i) = \begin{cases} logit^{-1}(\alpha + \beta x_i) \text{ if } y_i = 1 \\ 1 - logit^{-1}(\alpha + \beta x_i) \text{ if } y_i = 0 \end{cases}$$

$$= (logit^{-1}(\alpha + \beta x_i))^{y_i}(1 - logit^{-1}(\alpha + \beta x_i))^{1-y_i}$$

Logistic Regression

So we can define

$$L(\alpha, \beta) = \prod_i (logit^{-1}(\alpha + \beta x_i))^{y_i} (1 - logit^{-1}(\alpha + \beta x_i))^{1-y_i}$$

And try to maximize this quantity!

Unfortunately, there is no closed form solution here and we need to use numerical approximation methods to solve this optimization problem

Demo

Evaluating Our Regression Model

Some Notation:

$\mathbf{y_i}$ is the "true" value from our data set (i.e. $\mathbf{x_i}\boldsymbol{\beta} + \boldsymbol{\epsilon_i}$)

$\hat{\mathbf{y}}_\mathbf{i}$ is the estimate of y_i from our model (i.e. $\mathbf{x_i}\boldsymbol{\beta}_{\mathbf{LS}}$)

$\bar{\mathbf{y}}$ is the sample mean all $\mathbf{y_i}$

$\mathbf{y_i} - \hat{\mathbf{y}}_\mathbf{i}$ are the estimates of $\boldsymbol{\epsilon_i}$ and are referred to as residuals

Evaluating Our Regression Model

$$TSS = \sum_i^n (y_i - \bar{y})^2$$

$$RSS = \sum_i^n (y_i - \hat{y}_i)^2$$

$$ESS = \sum_i^n (\hat{y}_i - \bar{y})^2$$

$$R^2 = \frac{ESS}{TSS} = 1 - \frac{RSS}{TSS}$$

R^2 measures the fraction of variance that is explained by $\hat{y}$

Exercise

Assume for simplicity that $\hat{y}_i = \beta_0 + \beta_1 x_i$
Since β_0 and β_1 are least squares estimates, we know they minimize

$$\sum_i (y_i - \hat{y}_i)^2$$

By taking derivatives of the above with respect to β_0 and β_1 we discover that

$$\sum_i (y_i - \hat{y}_i) = 0 \text{ and } \sum_i (y_i - \hat{y}_i) x_i = 0$$

Show that TSS = ESS + RSS

$$\begin{aligned}
\mathrm{TSS} &= \sum_i (y_i - \bar{y})^2 \\
&= \sum_i (y_i - \hat{y}_i + \hat{y}_i - \bar{y})^2 \\
&= \sum_i (y_i - \hat{y}_i)^2 + \sum_i (\hat{y}_i - \bar{y})^2 + 2 \sum_i (y_i - \hat{y}_i)(\hat{y}_i - \bar{y}) \\
&= \mathrm{ESS} + \mathrm{RSS} + 2 \sum_i (y_i - \hat{y}_i)(\hat{y}_i - \bar{y}).
\end{aligned}$$

$$\begin{aligned}
\sum_i (y_i - \hat{y}_i)(\hat{y}_i - \bar{y}) &= \sum_i (y_i - \hat{y}_i)\hat{y}_i - \bar{y} \sum_i (y_i - \hat{y}_i) \\
&= \hat{\beta}_0 \sum_i (y_i - \hat{y}_i) + \hat{\beta}_1 \sum_i (y_i - \hat{y}_i) x_i - \bar{y} \sum_i (y_i - \hat{y}_i)
\end{aligned}$$

Evaluating our Regression Model

Each parameter of an independent variable **x** has an associated confidence interval

If the parameter / coefficient is not significantly distinguishable from 0 then we cannot assume that there is a significant linear relationship between that independent variable and the observations **y** (i.e. if the interval includes 0)

Confidence Intervals

How do we build a confidence interval?

Assume $\mathbf{Y_i \sim N(5, 25)}$, for $\mathbf{1 \le i \le 100}$ and $\mathbf{y_i = \mu + \epsilon}$ where $\mathbf{\epsilon \sim N(0, 25)}$. Then the Least Squares estimator of $\mathbf{\mu}$ ($\mathbf{\mu_{LS}}$) is

the sample mean $\mathbf{\bar{y}}$

What is the 95% confidence interval for $\mathbf{\mu_{LS}}$?

$$\mathbf{SE(\mu_{LS}) = \sigma_\epsilon / \sqrt{n} = 5 / \sqrt{100} = .5}$$

$$\mathbf{CI_{.95} = [\bar{y} - 1.96 \times SE(\mu_{LS}), \bar{y} + 1.96 \times SE(\mu_{LS})]}$$
$$\mathbf{= [\bar{y} - 1.96 \times .5, \bar{y} + 1.96 \times .5]}$$

Z-value for 95% Confidence Interval

Z-values

These are the number of standard deviations from the mean of a N(0,1) distribution required in order to contain a specific % of values were you to sample a large number of times.

To find the .95 z-value (the number of standard deviations from the mean that contains 95% of values) you need to solve:

$$\int_{-z}^{z} \frac{1}{2\pi} e^{-\frac{1}{2}x^2} dx = .95$$

QQ plot

We need to check our assumption that our residuals / noise estimates are normally distributed.

How do can you check that a variable follows a specific distribution?

Need to check that our variable is **distributed** in the same way that a variable following our target distribution would be.

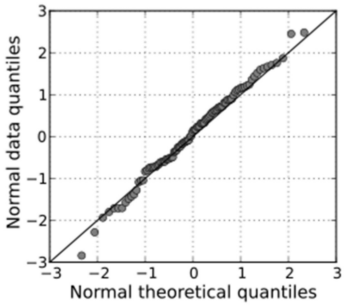
Plot the quantile of your target distribution against the quantiles of your data/ variable! If they match then your data probably comes from that distribution.

QQ plot

Quantiles are the values for which a particular % of values are contained below it.

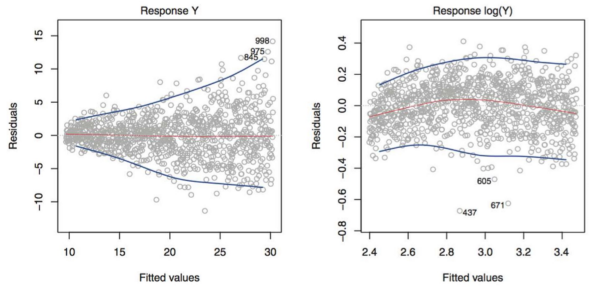
For example the 50% quantile of a N(0,1) distribution is 0 since 50% of samples would be contained below 0 were you to sample a large number of times.

QQ plot



Constant Variance

One of our assumptions was that our noise had constant variance. How can we verify this?
We can plot our fitted values against our residuals (noise estimates)



Extending our Linear Model

Changing the assumptions we made can drastically change the problem we are solving. A few ways to extend the linear model:

1. Non-constant variance - used in WLS (weighted least squares)
2. Distribution of error is not Normal - used in GLM (generalized linear models)

File - D:\Users\Wolfs\Documents\GitHub\CS506-Fall2021\10-regression\linear-regression.py

```python
import numpy as np
import pandas as pd
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from sklearn import datasets
import statsmodels.api as sm

import seaborn as sns; sns.set()

# We generate our data
# Then apply our transfrom of X and X transpose to the data
# If you think X2 may be a term, you need to include it in the terms you try

# plot line
# We know what the parameters are Beta0 is 1, Beta1 is 0.5
line = np.array([1, 0.5])
xlin = -10.0 + 20.0 * np.random.random(100)
ylin = line[0]+(line[1]*xlin)+np.random.randn(100)
plt.plot(xlin,ylin,'ro',markersize=4)
plt.show()

# plot quadratic
# There are ways to explore your data so you can decide what parameters to try
quad = np.array([1, 3, 0.5])
xquad = -10.0 + 20.0 * np.random.random(100)
# Calling it xquad is a little misleading
# How did you know if was X2? That's where being a data scientist comes in
yquad = quad[0]+(quad[1]*xquad)+(quad[2]*xquad*xquad)+np.random.randn(100)
plt.plot(xquad,yquad,'ro',markersize=4)
plt.show()

# plot log
# You can see that shape is clearly logarithmic so you need to include a log term
# If we get negative points, then our model would bbe
```

```
clearly wrong and it wasn't actually log
# If you repeat this many times, you expect on average
    you will get the true values you had generated
log = np.array([1, 4])
xlog = 10.0 * np.random.random(100)
ylog = log[0]+log[1]*np.log(xlog)+np.random.randn(100)
plt.plot(xlog,ylog,'ro',markersize=4)
plt.show()

# plot line through first plot
line = np.array([1, 0.5])
xlin = -10.0 + 20.0 * np.random.random(100)
ylin = line[0]+(line[1]*xlin)+np.random.randn(100)
plt.plot(xlin,ylin,'ro',markersize=4)
plt.plot(xlin,line[0]+line[1]*xlin,'b-')
plt.text(-9,3,r'$y = \beta_0 + \beta_1x$',size=20)
plt.show()

# Least square estimate through first plot
m = np.shape(xlin)[0]
X = np.array([np.ones(m),xlin]).T
beta = np.linalg.inv(X.T @ X) @ X.T @ ylin

xplot = np.linspace(-10,10,50)
yestplot = beta[0]+beta[1]*xplot
plt.plot(xplot,yestplot,'b-',lw=2)
plt.plot(xlin,ylin,'ro',markersize=4)
plt.show()
print(beta)

# Least square estimate through second plot
m = np.shape(xquad)[0]
X = np.array([np.ones(m),xquad,xquad**2]).T
beta = np.linalg.inv(X.T @ X) @ X.T @ yquad

xplot = np.linspace(-10,10,50)
yestplot = beta[0]+beta[1]*xplot+beta[2]*xplot**2
plt.plot(xplot,yestplot,'b-',lw=2)
plt.plot(xquad,yquad,'ro',markersize=4)
plt.show()
print(beta)
```

```
 74
 75 # Least square estimate through third plot
 76 m = np.shape(xlog)[0]
 77 X = np.array([np.ones(m),np.log(xlog)]).T
 78 beta = np.linalg.inv(X.T @ X) @ X.T @ ylog
 79
 80 xplot = np.linspace(-10,10,50)
 81 yestplot = beta[0]+beta[1]*np.log(xplot)
 82 plt.plot(xplot,yestplot,'b-',lw=2)
 83 plt.plot(xlog,ylog,'ro',markersize=4)
 84 plt.show()
 85 print(beta)
 86
 87 # using libraries
 88 # There are libraries to do this for you
 89 # We can plot this in 3D (pycharm showed it flat, but
     there may be a way to manipulate it)
 90 X, y = datasets.make_regression(n_samples=100,
    n_features=2, n_informative=5, noise=30, random_state
    =1)
 91 X = sm.add_constant(X)
 92 ax = plt.axes(projection='3d')
 93 ax.scatter3D(X.T[1], X.T[2], y, cmap='ro')
 94 plt.show()
 95
 96 model = sm.OLS(y, X)
 97 results = model.fit()
 98
 99 ax = plt.axes(projection='3d')
100 ax.scatter3D(X.T[1], X.T[2], y)
101
102 x1, x2 = np.meshgrid(np.arange(min(X.T[1]), max(X.T[1
    ]), .5), np.arange(min(X.T[2]), max(X.T[2]), .5))
103 exog = pd.core.frame.DataFrame({'x0': np.ones(len(x1.
    ravel())).ravel(), 'x1': x1.ravel(), 'x2':x2.ravel
    ()})
104 print(exog)
105 out = results.predict(exog=exog)
106 ax.plot_surface(x1, x2, out.values.reshape(x1.shape
    ), color='Green', alpha=.2)
107 plt.show()
```

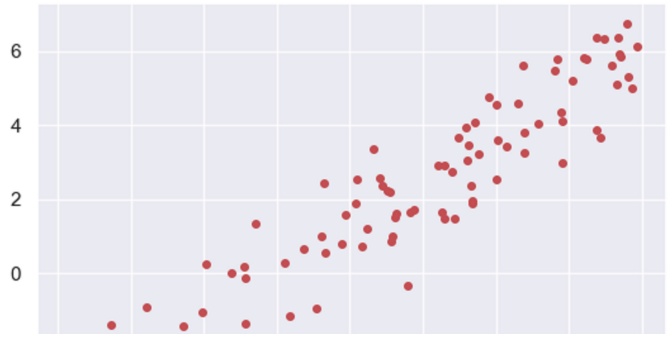
File - D:\Users\Wolfs\Documents\GitHub\CS506-Fall2021\10-regression\linear-regression.py

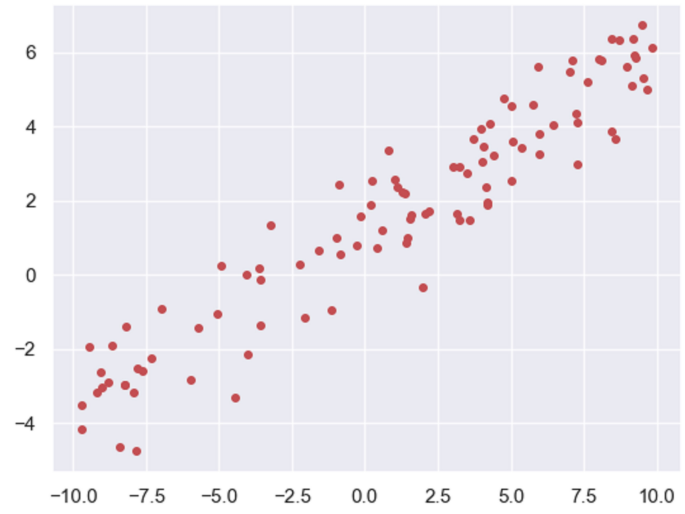
```
108
109 # Evaluating the Modal
110 print(results.summary())
111
```

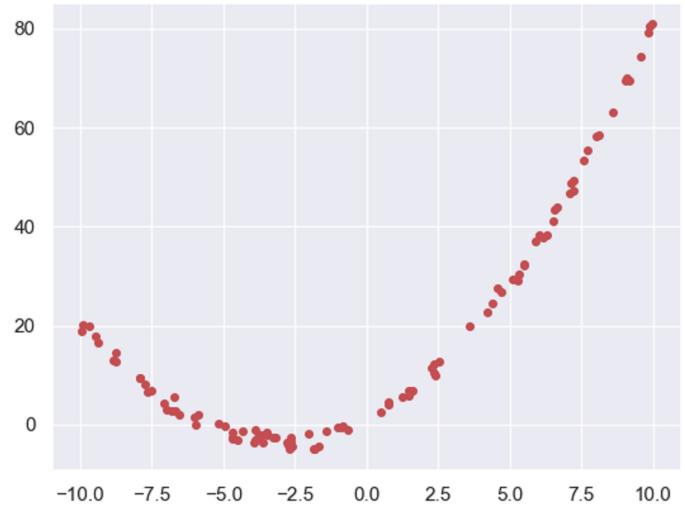


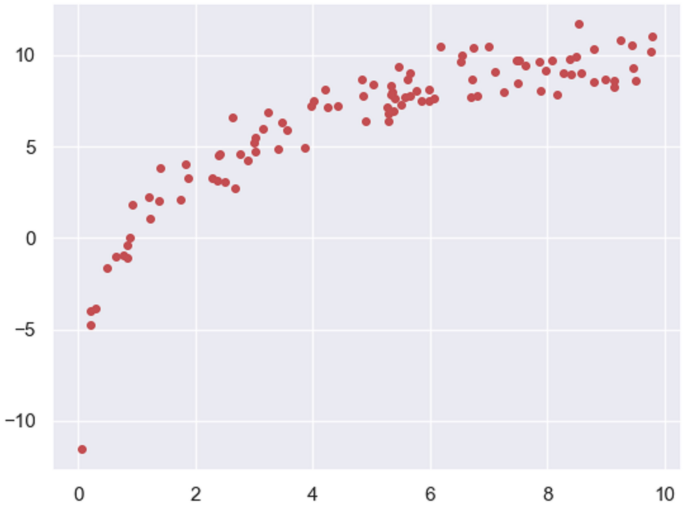
```
./linear-regression.py
[1.01858886 0.51037431]
[1.24622978 3.00077925 0.49682696]
D:\Users\Wolfs\Documents\GitHub\CS506-Fall2021\10-regression\linear-regression.py:70: RuntimeWarning:
invalid value encountered in log
  yestplot = beta[0]+beta[1]*np.log(xplot)
[0.94582328 4.0255589 ]
    x0       x1       x2
0   1.0 -2.434838 -1.857982
1   1.0 -1.934838 -1.857982
2   1.0 -1.434838 -1.857982
3   1.0 -0.934838 -1.857982
```

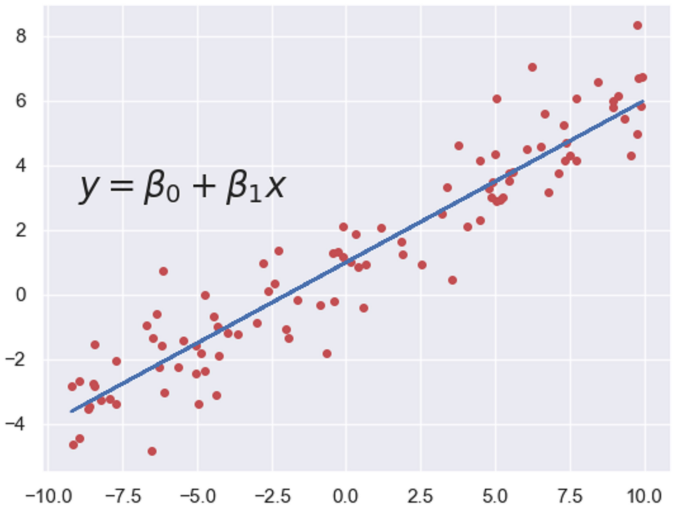
```
4   1.0 -0.434838 -1.857982
..  ...      ...      ...
85  1.0  0.065162  2.142018
86  1.0  0.565162  2.142018
87  1.0  1.065162  2.142018
88  1.0  1.565162  2.142018
89  1.0  2.065162  2.142018

[90 rows x 3 columns]
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.840
Model:                            OLS   Adj. R-squared:                  0.836
Method:                 Least Squares   F-statistic:                     254.1
Date:                Wed, 20 Oct 2021   Prob (F-statistic):           2.72e-39
Time:                        17:35:38   Log-Likelihood:                -482.37
No. Observations:                 100   AIC:                             970.7
Df Residuals:                      97   BIC:                             978.5
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.1912      3.162      0.693      0.490      -4.085       8.467
x1            29.3912      3.274      8.977      0.000      22.893      35.889
x2            78.1391      3.594     21.741      0.000      71.006      85.272
==============================================================================
Omnibus:                        1.279   Durbin-Watson:                   1.824
Prob(Omnibus):                  0.527   Jarque-Bera (JB):                1.065
Skew:                           0.253   Prob(JB):                        0.587
Kurtosis:                       2.999   Cond. No.                         1.38
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Process finished with exit code 0
```

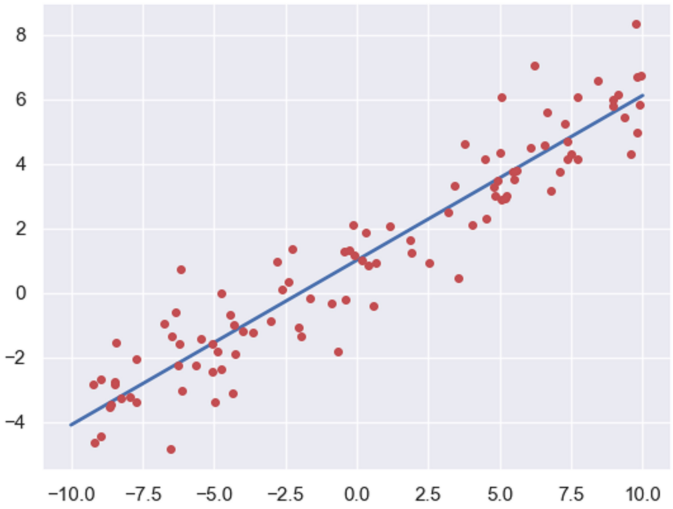


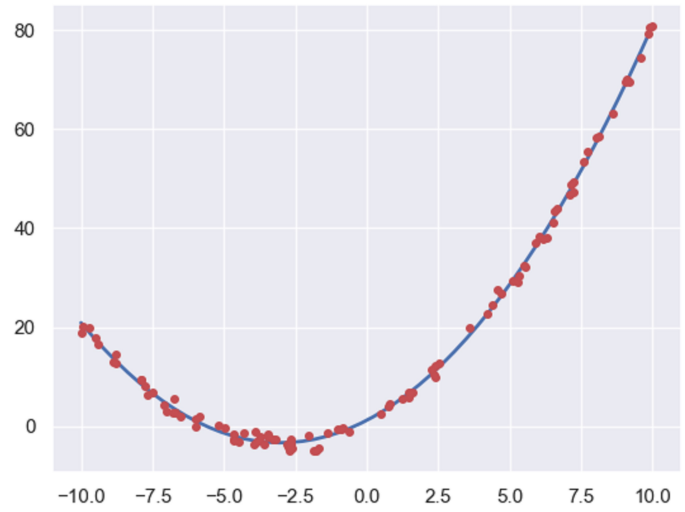

6
4
2
0
−2
−4
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

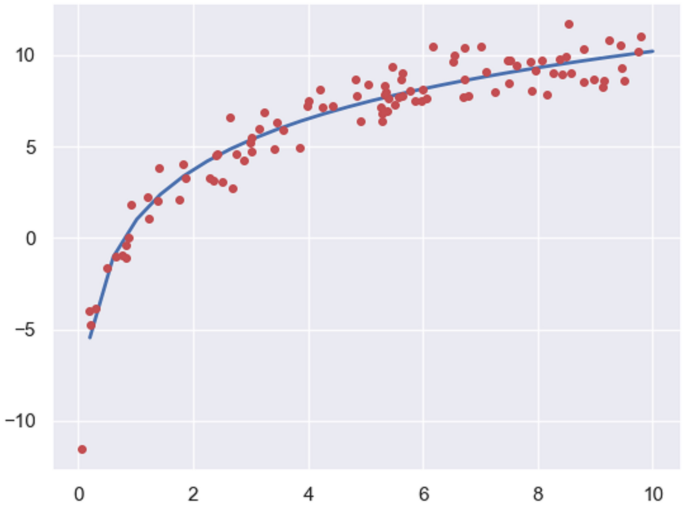

80
60
40
20
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

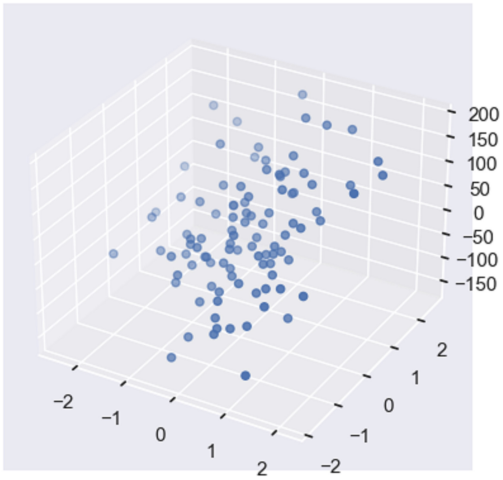

10
5
0
−5
−10
0
2
4
6
8
10

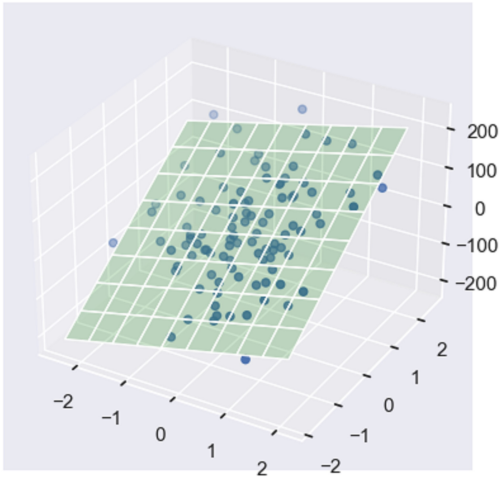

$y = \beta_0 + \beta_1 x$
8
6
4
2
0
−2
−4
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


8
6
4
2
0
−2
−4
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


80
60
40
20
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


10
5
0
−5
−10
0
2
4
6
8
10


200
150
100
50
0
−50
−100
−150
−2
−1
0
1
2
2
1
0
−1
−2


200
100
0
−100
−200
−2
−1
0
1
2
−2
−1
0
1
2