Linear and Logistic Models

Interpreting $h(x)$

Boston House Prices

$$\theta = \begin{bmatrix} 1 \\ 164 \\ 1.25 \end{bmatrix}$$

x_1: floor Area

x_2: number of floors

$h(X)$: predicted price of the house

Boston House Prices

Between x_1 and x_2, x_1 is the value with higher magnitude. This means that x_1 contributes more in increasing the value of theta. Therefore, between **floor area** and **number of floors**, **floor area** matters more when a house is priced.

Iris Flowers Dataset

$$\theta = \begin{bmatrix} -1.9 & -1 & -3 \\ -1.2 & 0.2 & -0.01 \\ 2 & -1.4 & -0.16 \\ -2 & 0.8 & 2.1 \\ -1.9 & -0.9 & 3.1 \end{bmatrix}$$

Iris Flowers Dataset

x_1: sepal length

x_2: sepal width

x_3: petal length

x_4: petal width

$h_1(X)$: $P(\text{species} = \text{Setosa}|X)$

$h_2(X)$: $P(\text{species} = \text{Versicolor}|X)$

$h_3(X)$: $P(\text{species} = \text{Virginica}|X)$

Thetas in Regression

$$\theta = \begin{bmatrix} 5 \\ -100 \\ 100 \end{bmatrix}$$

positive θ values contribute to the increase of the value of the prediction while negative θ values contribute to the decrease of the value of the prediction

Thetas in Classification

$$\theta = \begin{bmatrix} 5 \\ -100 \\ 100 \end{bmatrix}$$

positive θ values contribute to the increase of the probability that the model predicts 1 while negative θ values contribute to the decrease of the probability that the model predicts 1

Normal Equation

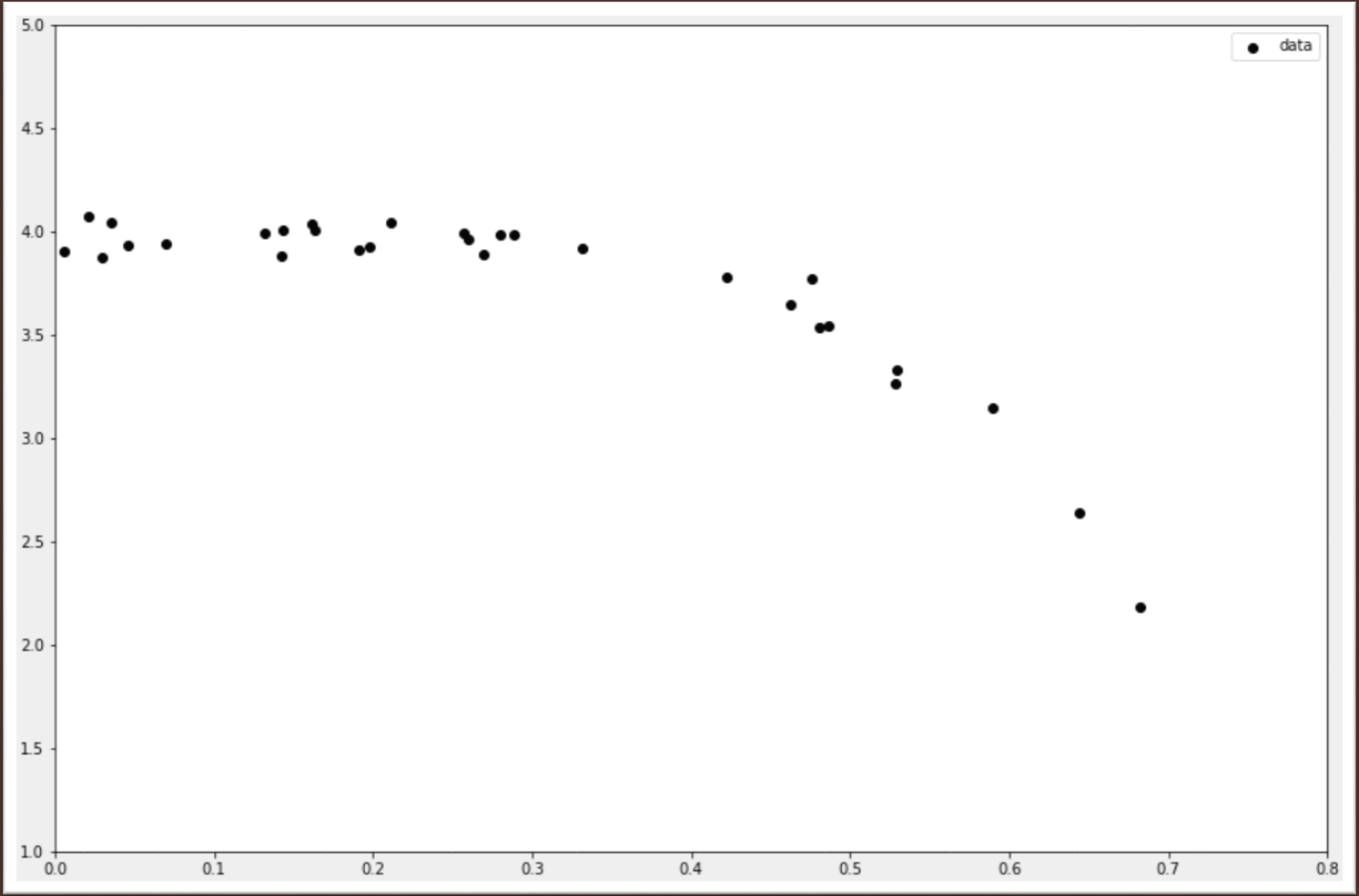
$$\theta = (X^T X)^{-1} X^T y$$

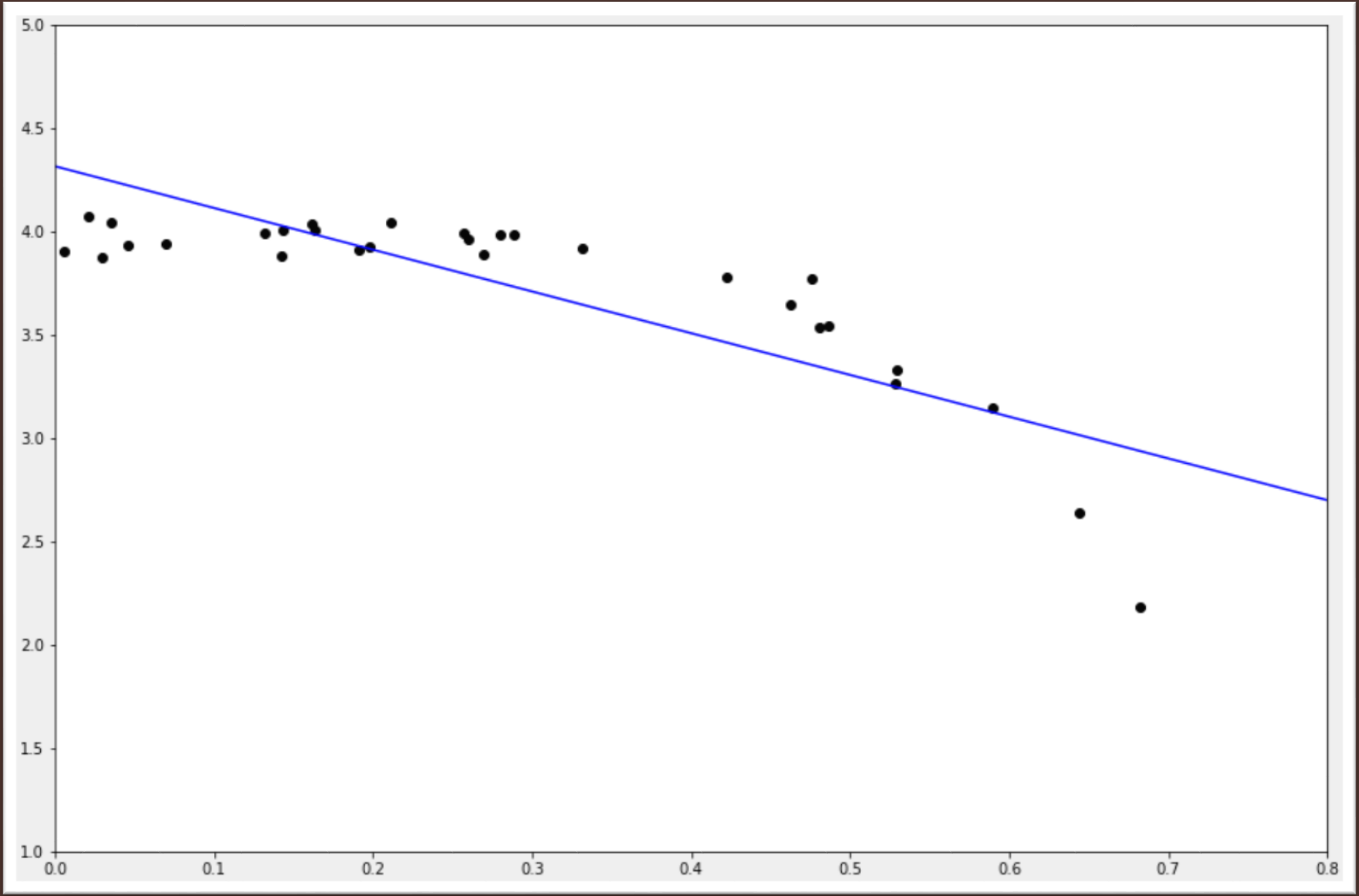
Normal Equation

- No need to choose value of α
- Matrix operations only
- Not susceptible to getting stuck on local minima
- Slow on large values of n

Gradient Descent

- Need to decide on the value of α
- Needs many iterations
- Susceptible to getting stuck in local minima
- Works well even on large values of n


data
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8


5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

Simple linear regression will always **UNDERFIT** this dataset since the actual relationship between the input and the response is too complex for an $h(x)$ straight line

Underfitting occurs when $h(x)$ is unable to generalize the relationship of the input and response due to insufficient or overly complex data

Polynomial Feature Engineering

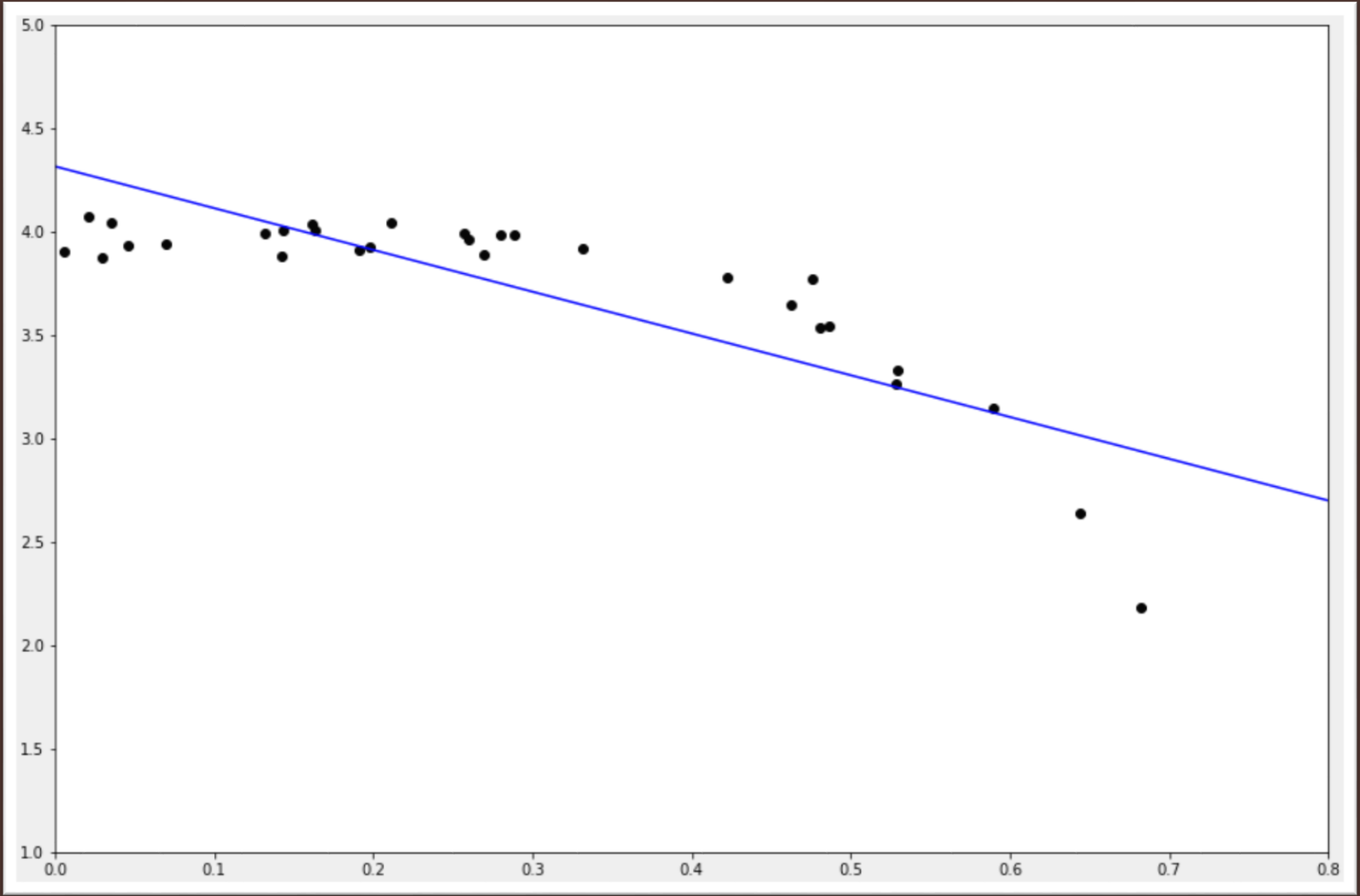
$$h_{lin}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

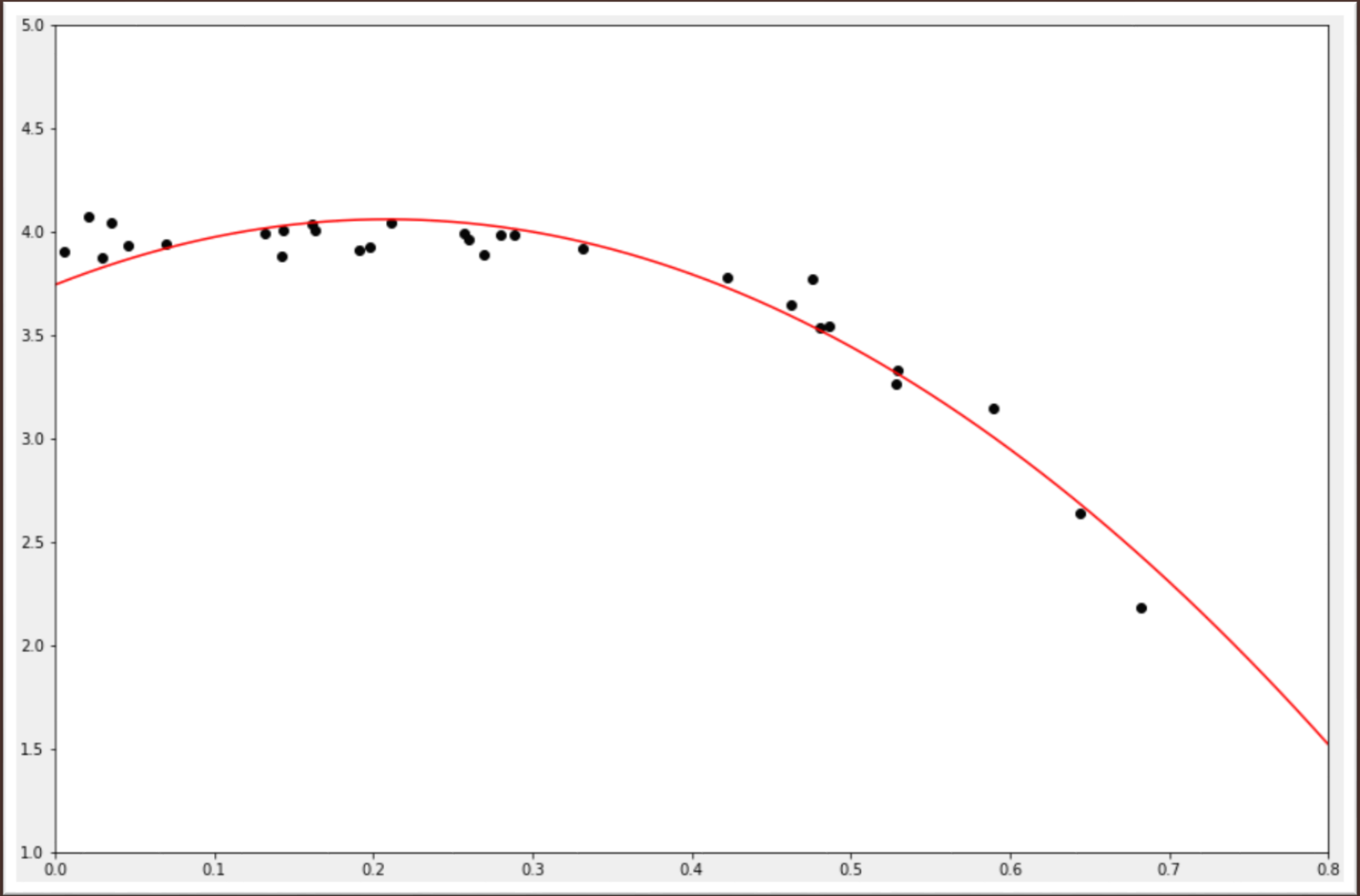
$$\begin{aligned} h_{lin}(x) \\ = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 \\ + \theta_4 x_1^2 + \theta_5 x_2^2 + \theta_6 x_1^2 x_2 + \theta_7 x_1 x_2^2 \\ + \theta_8 x_1^2 x_2^2 \end{aligned}$$

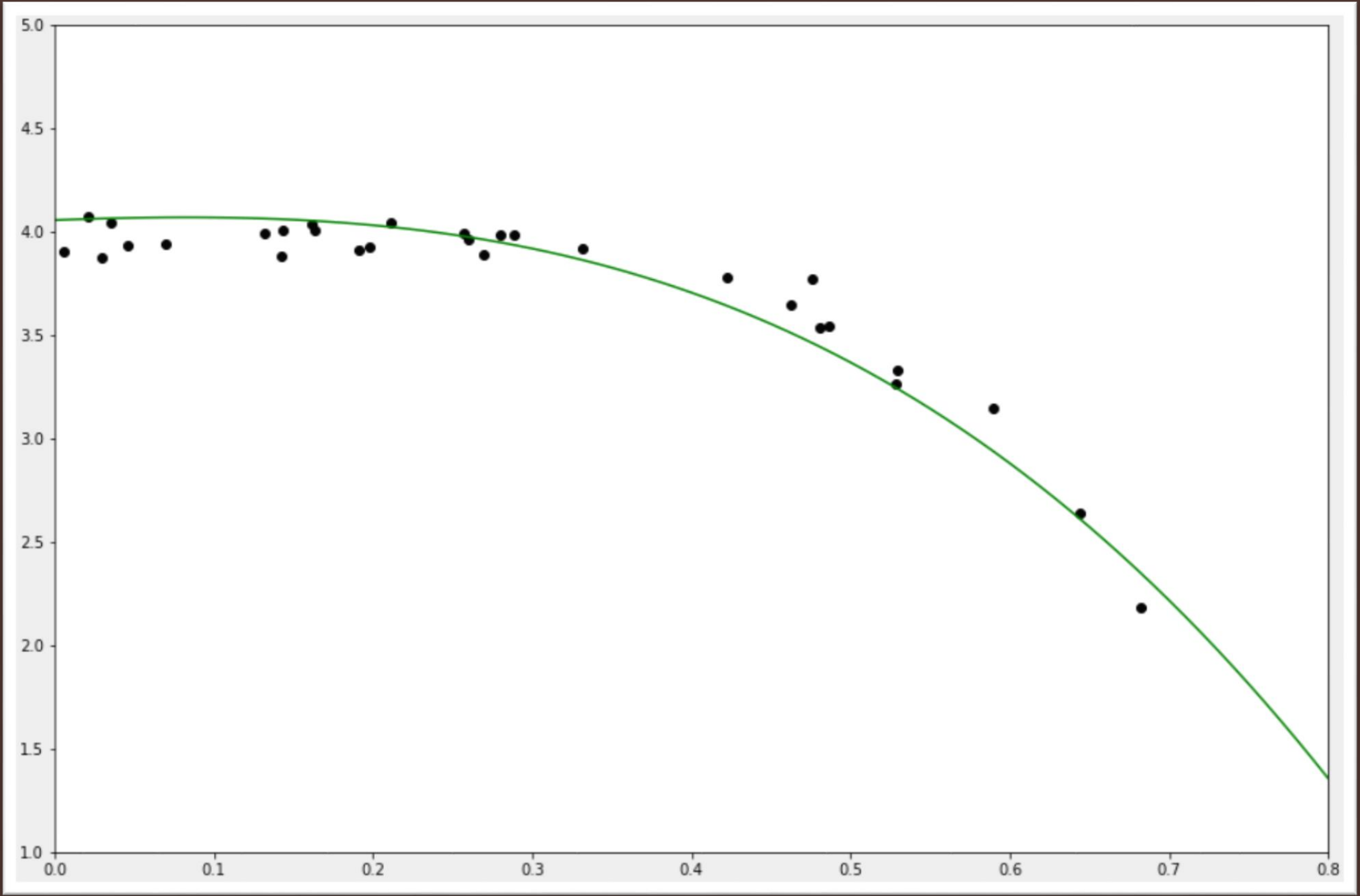
$$h_{lin}(x) = \theta_0 + \theta_1 x_1 + \theta_2 \sqrt{x_1}$$

x_0	x_1	x_2
1	1	3
1	4	1
1	3	3
1	2	2
1	1	1
1	4	5
1	6	1
1	2	1

$x_3 = x_1 x_2$	$x_4 = x_1^2$	$x_5 = x_2^2$	$x_6 = x_1^2 x_2$	$x_7 = x_1 x_2^2$	$x_8 = x_1^2 x_2^2$
3	1	9	3	9	9
4	16	1	16	4	16
9	9	9	27	27	81
4	4	4	8	8	16
1	1	1	1	1	1
20	16	25	80	100	400
6	36	1	36	6	36
2	4	1	4	2	4


5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8


5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

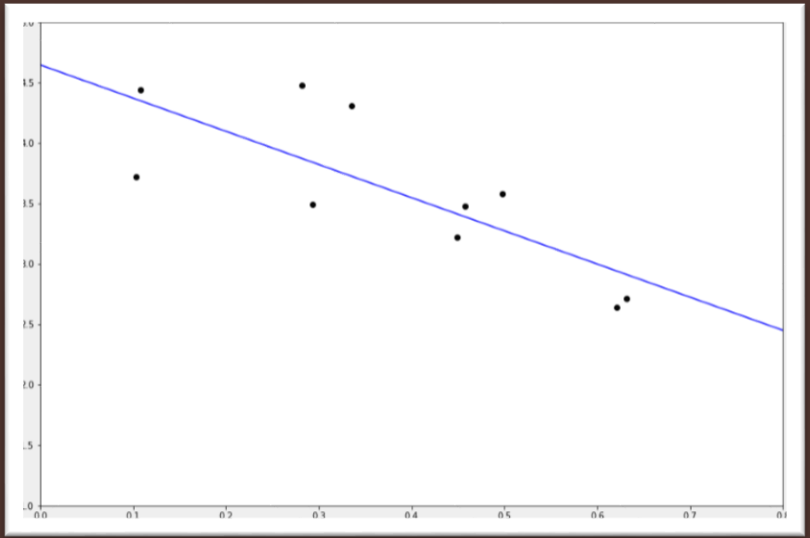
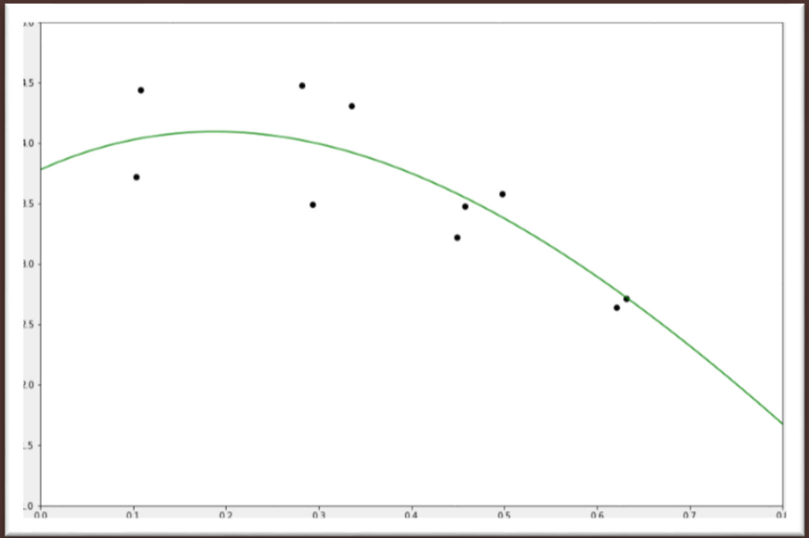
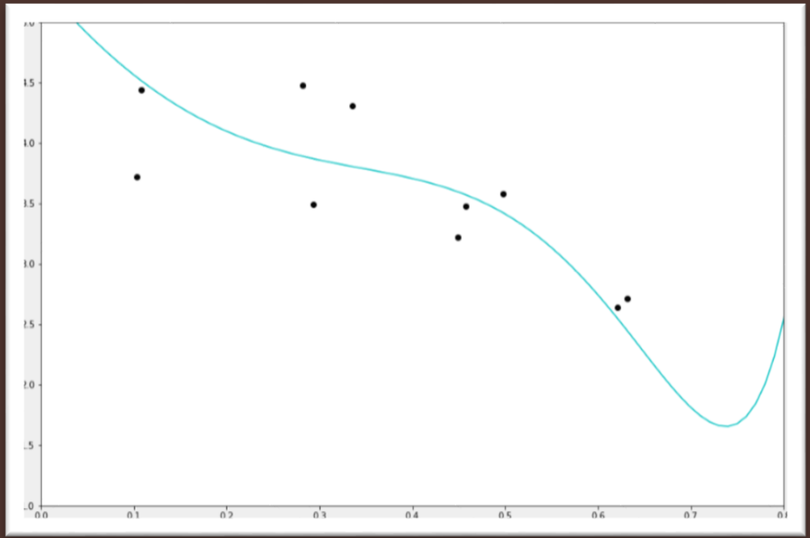
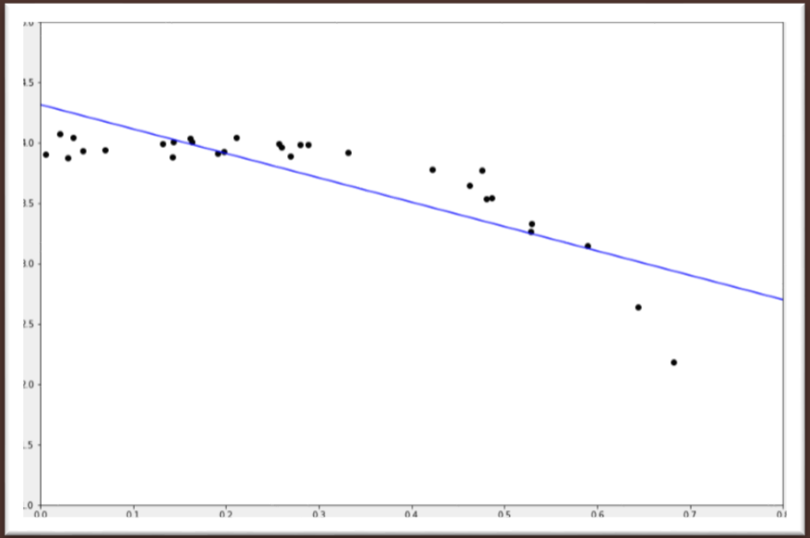
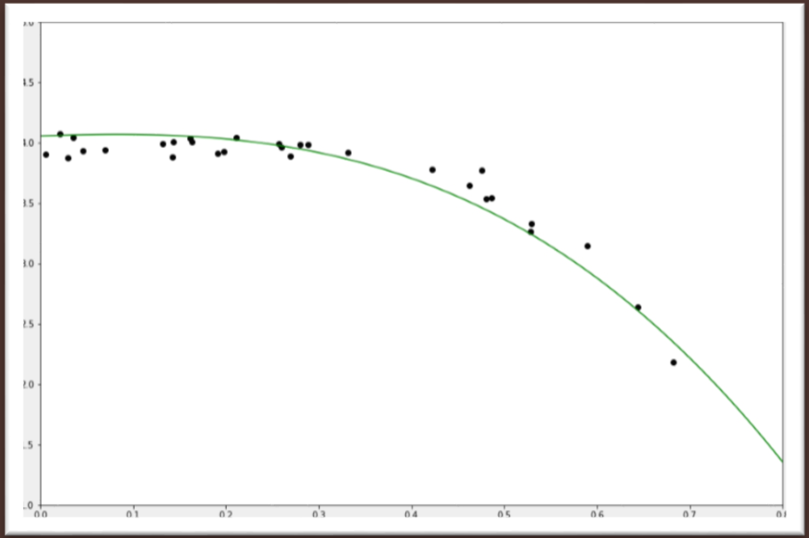
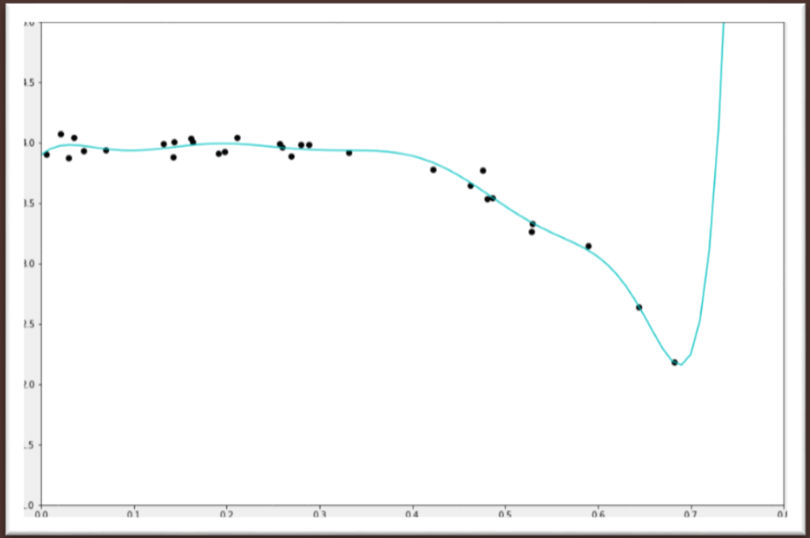

5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

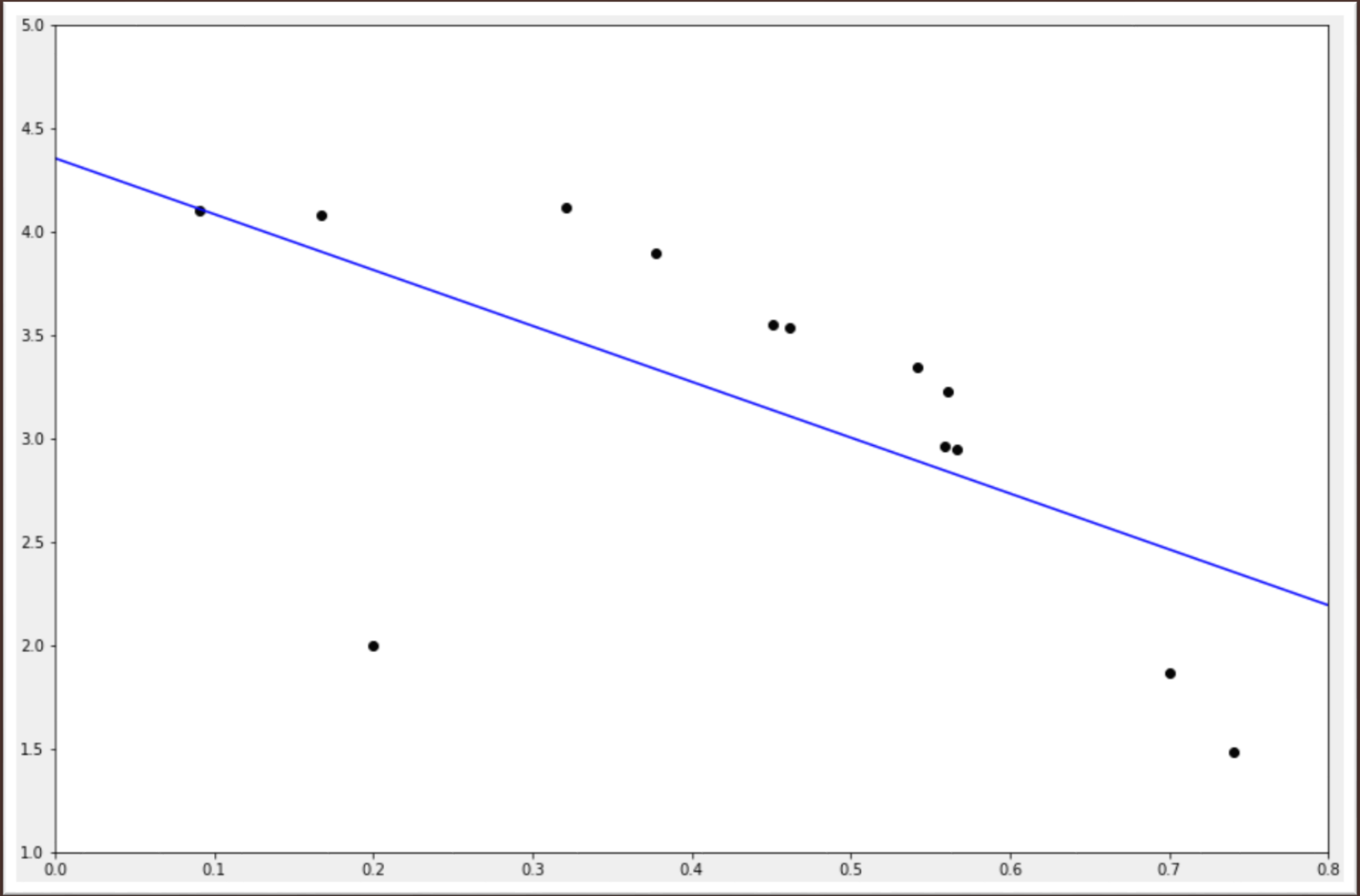
during these cases, feature scaling becomes more important since engineering features with large values using high polynomial degrees leads to extremely large values.

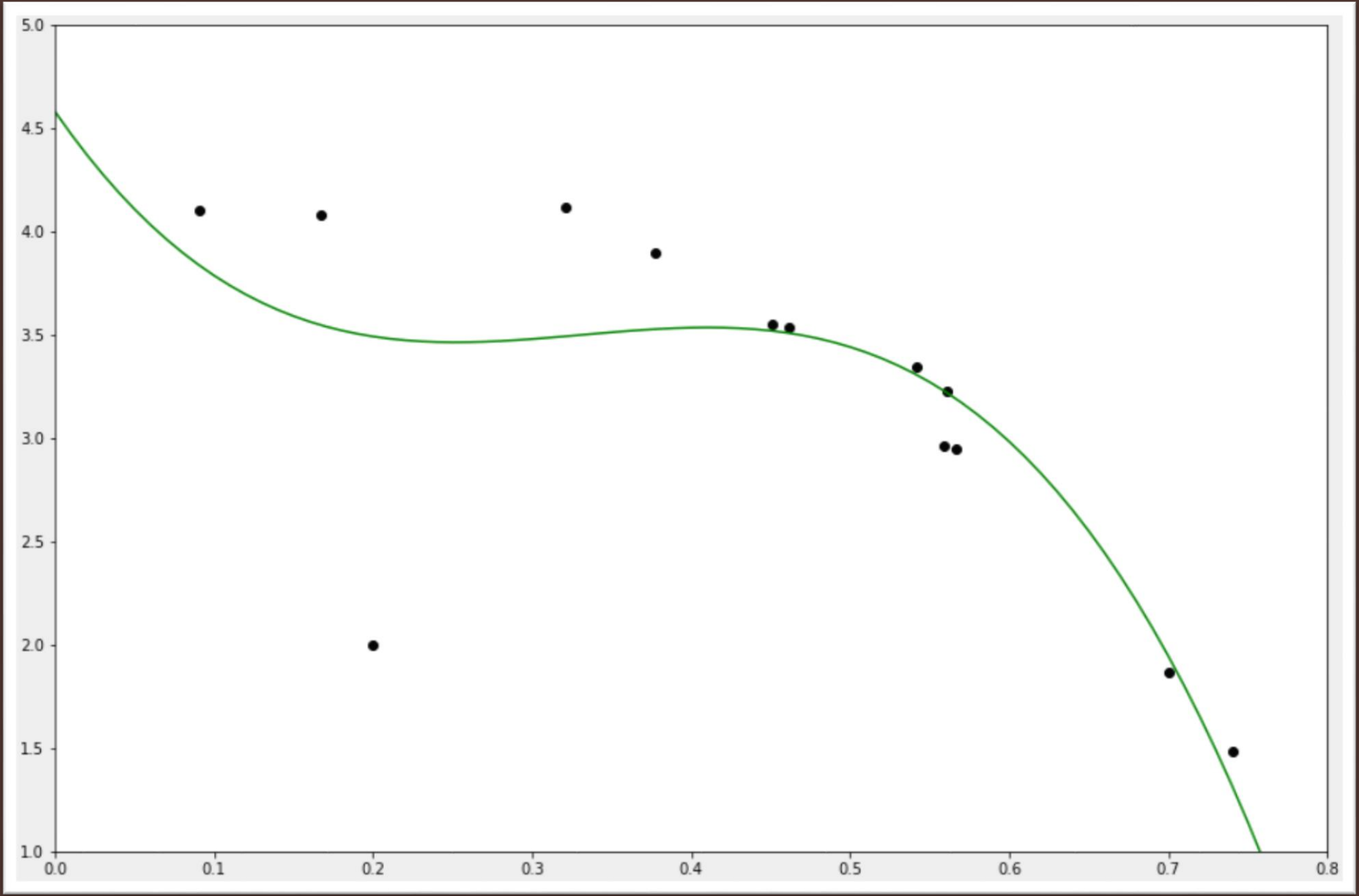
If x_1 has a range of $[1{,}1000]$, x_1^2 has a range $[1{,}1000000]$, and x_1^3 has a range $[1{,}1000000000]$

engineering more features will improve the fit of the $h(x)$ to the dataset

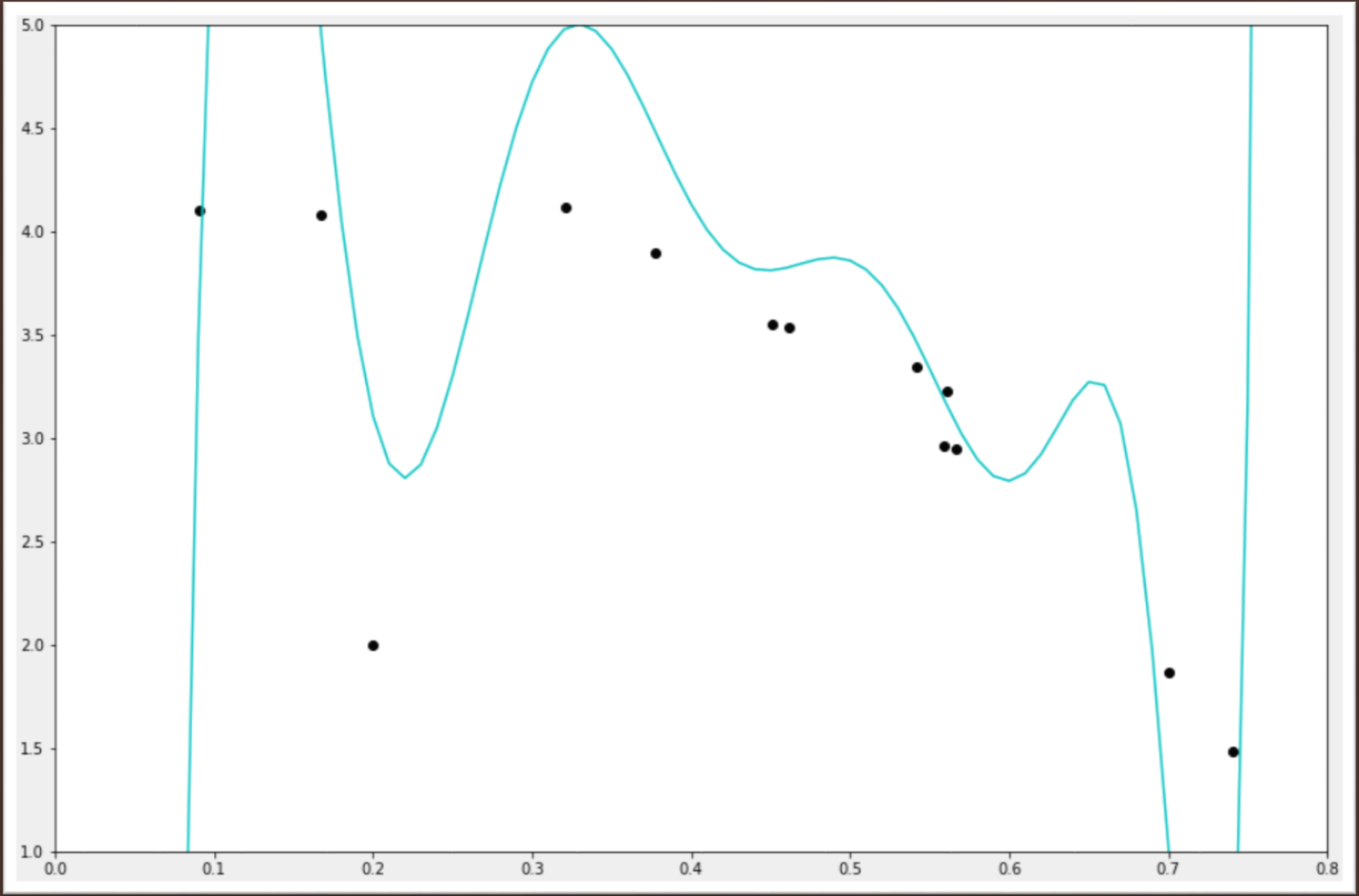
engineering too much features however, will **OVERFIT** $h(x)$ to the dataset

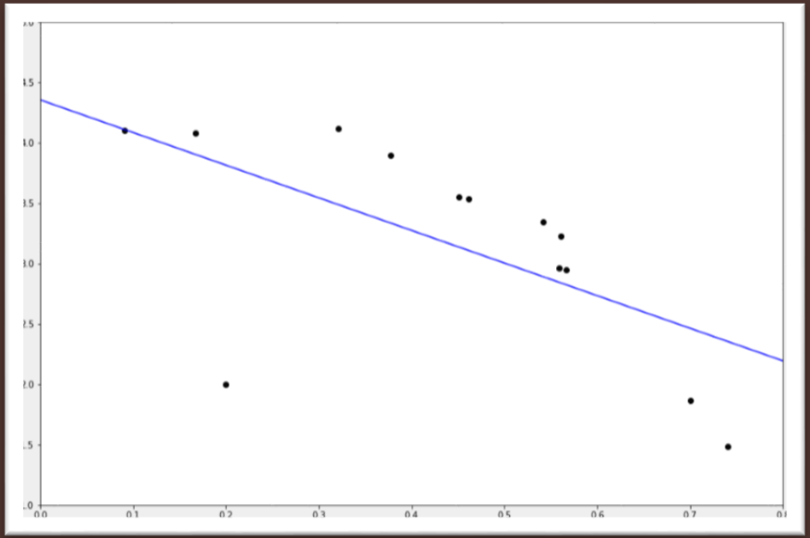







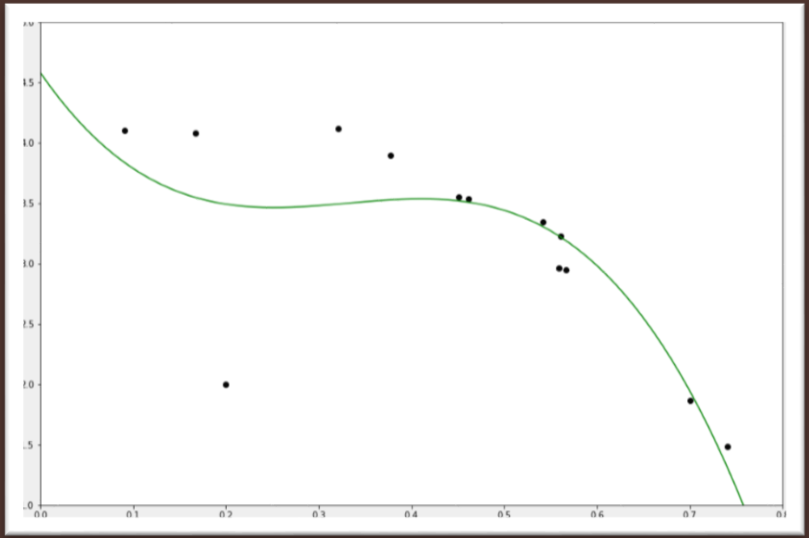

5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

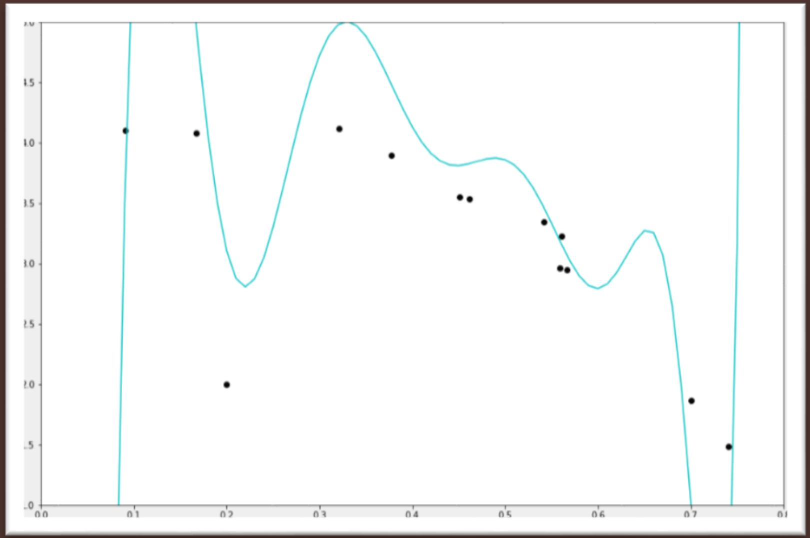


5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8


5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8







overfitting occurs when $h(x)$ models the noise and outliers of the dataset instead of modelling the actual relationships between the input and the response overfitting also means that the algorithm is overstating the influence of the values of theta

Addressing Overfitting

Addressing overfitting is one of the main concerns of machine learning. There are plenty of ways to reduce overfitting, the effectiveness of these ways depends on the problem

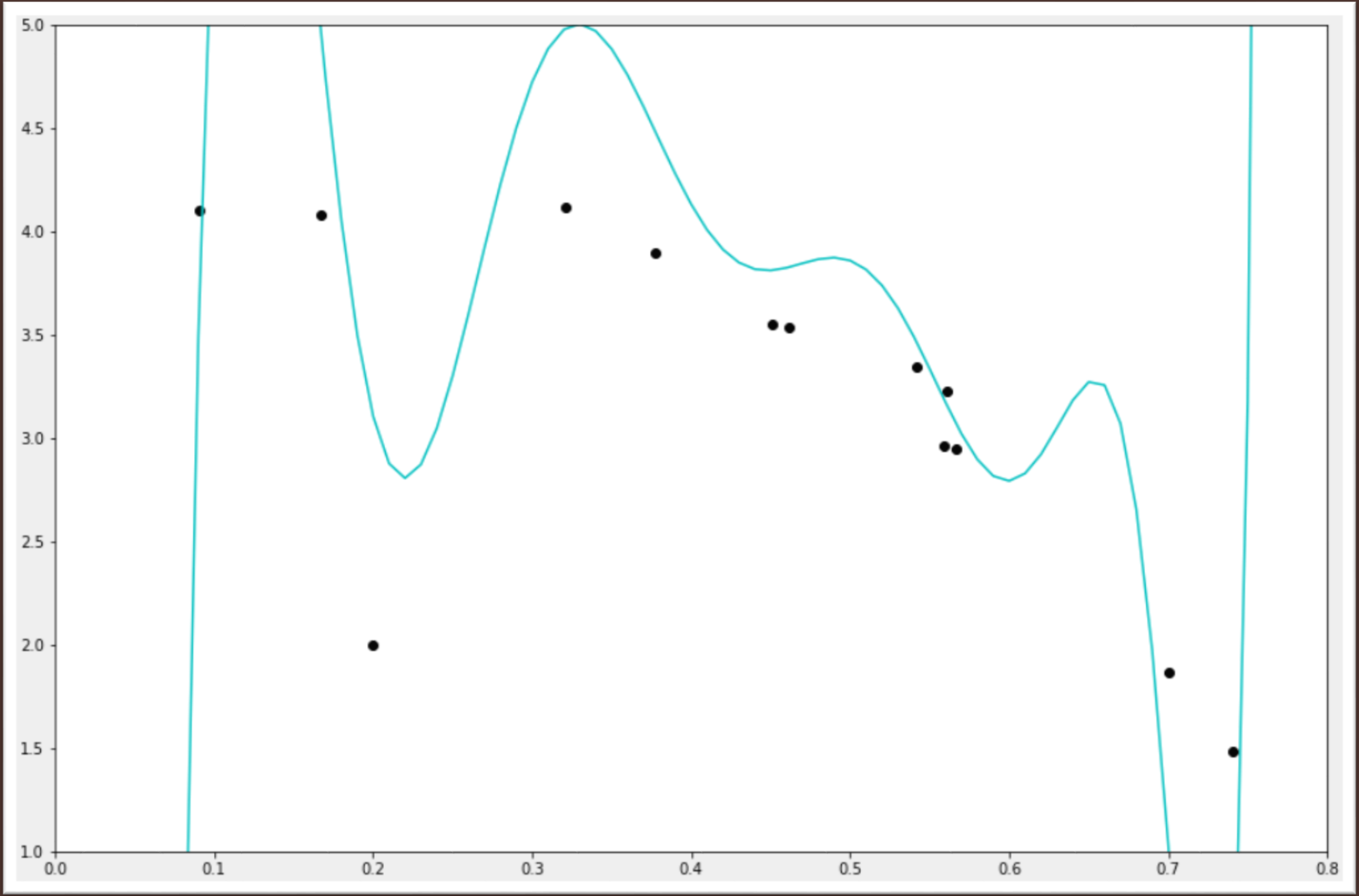
- Reducing the amount of features
- Regularization
- Other techniques specific to the algorithm

Regularization

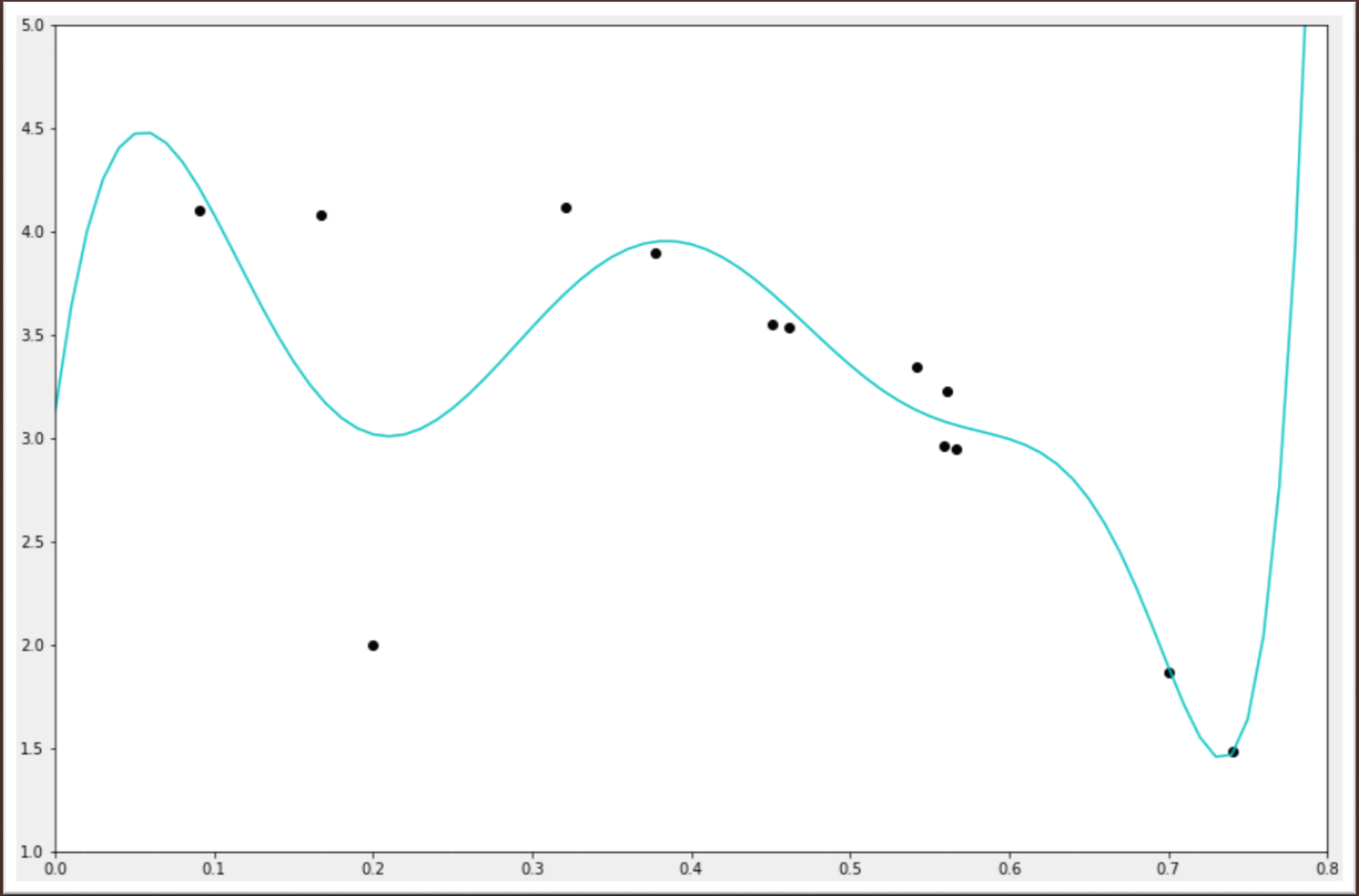
The objective of regularization is to reduce the influence of features with high degree

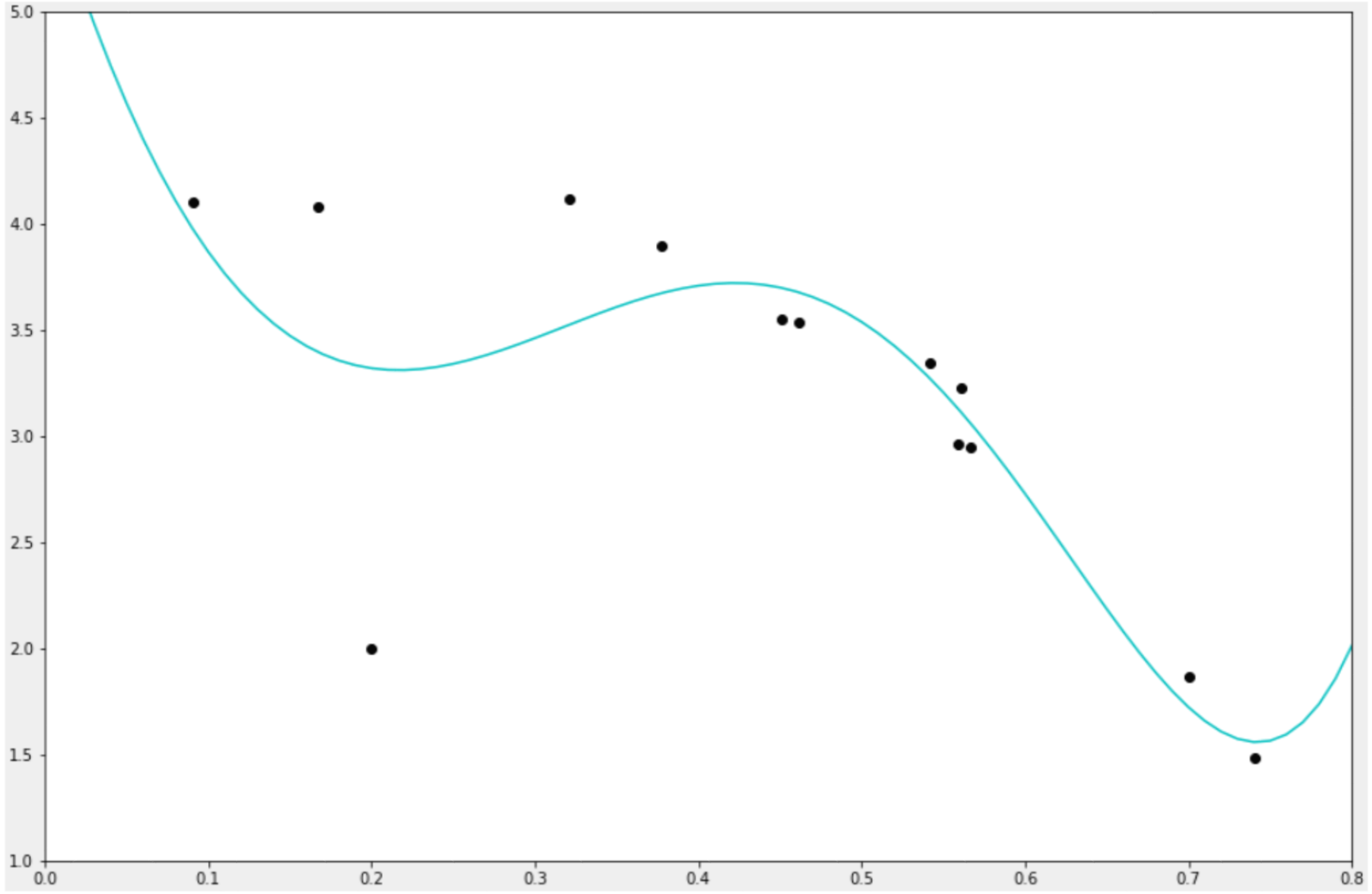
$$\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3 + \theta_4 x_1^4$$

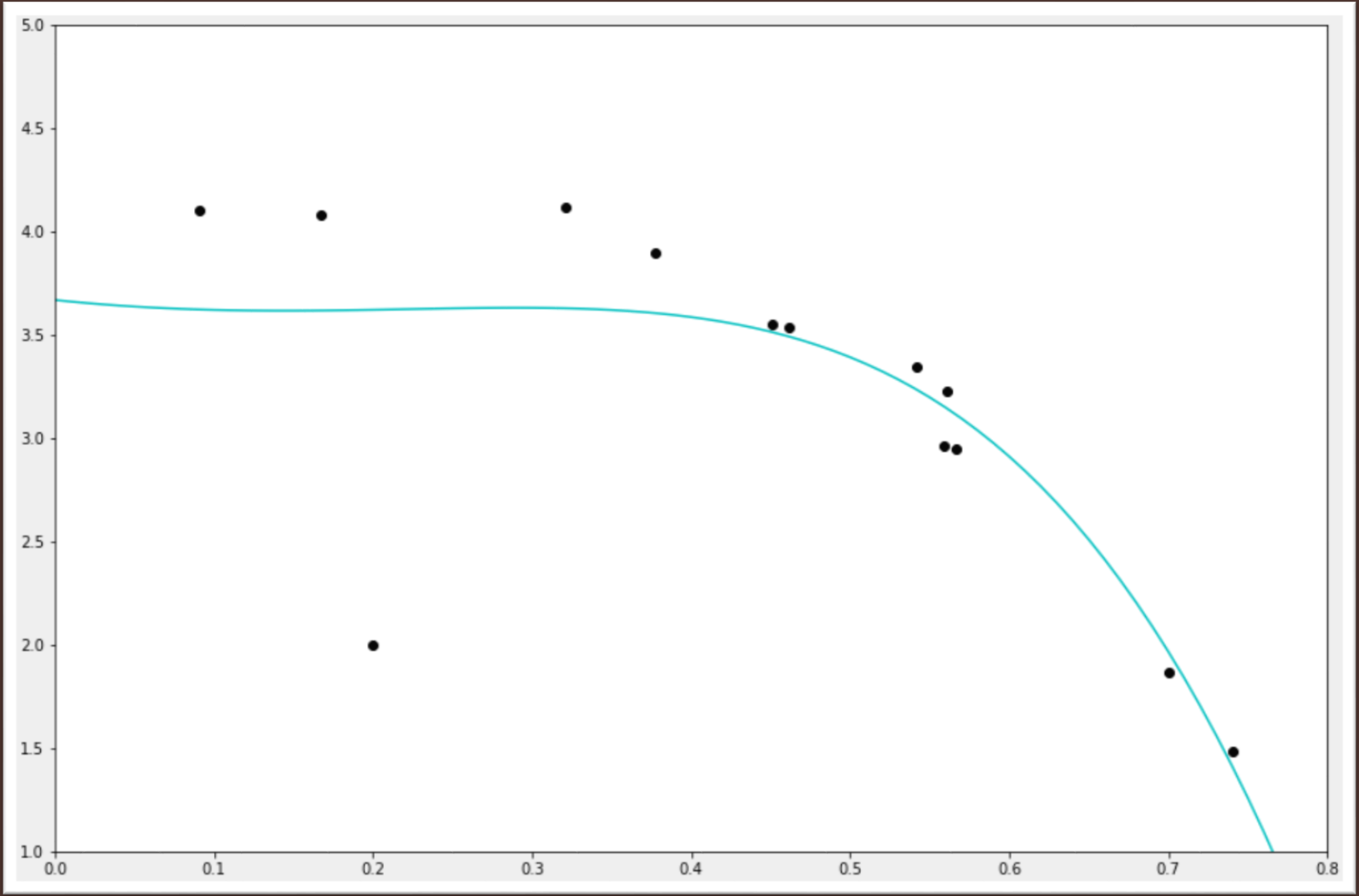
$$\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3 + \theta_4 x_1^4$$



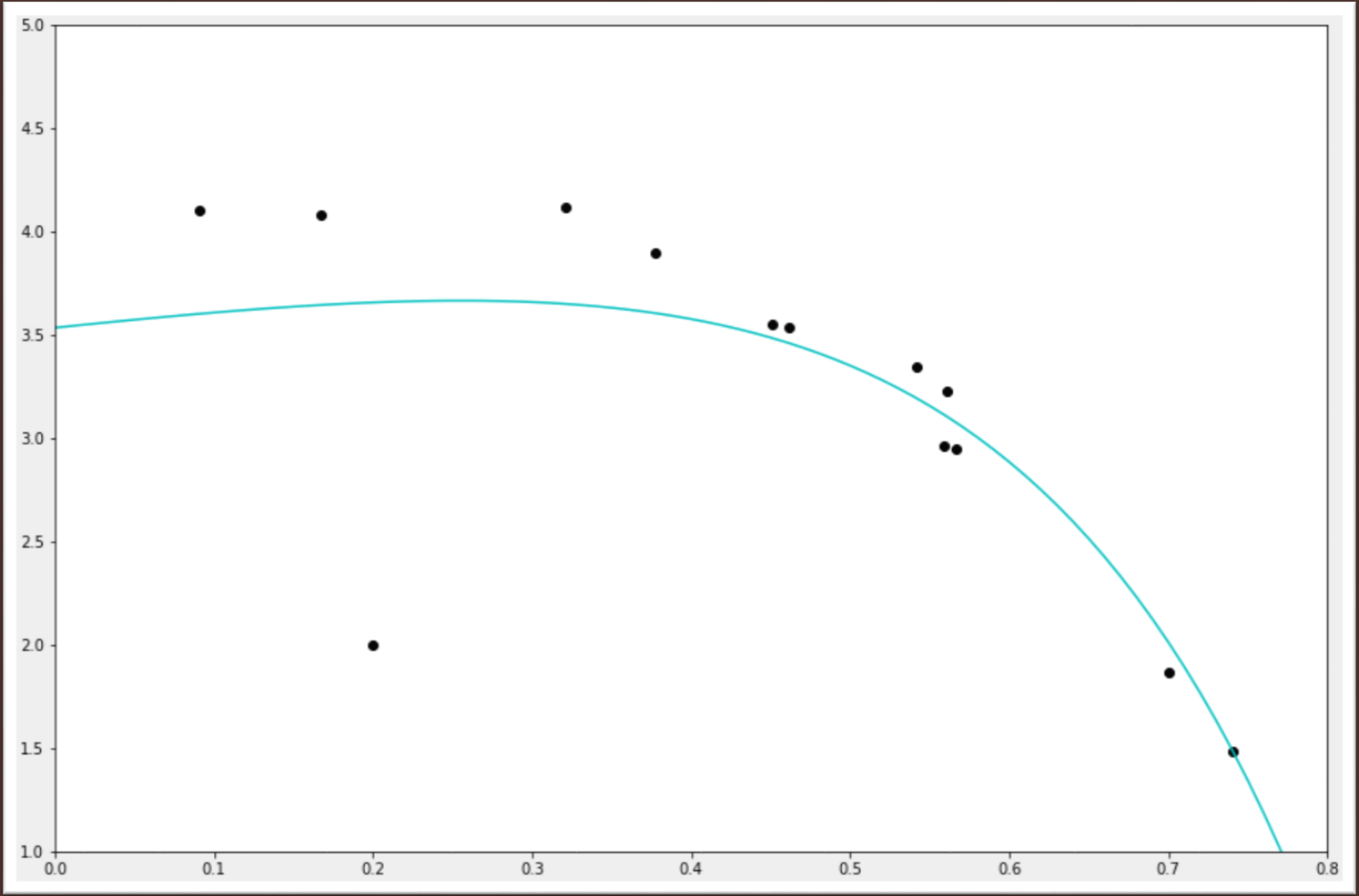
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

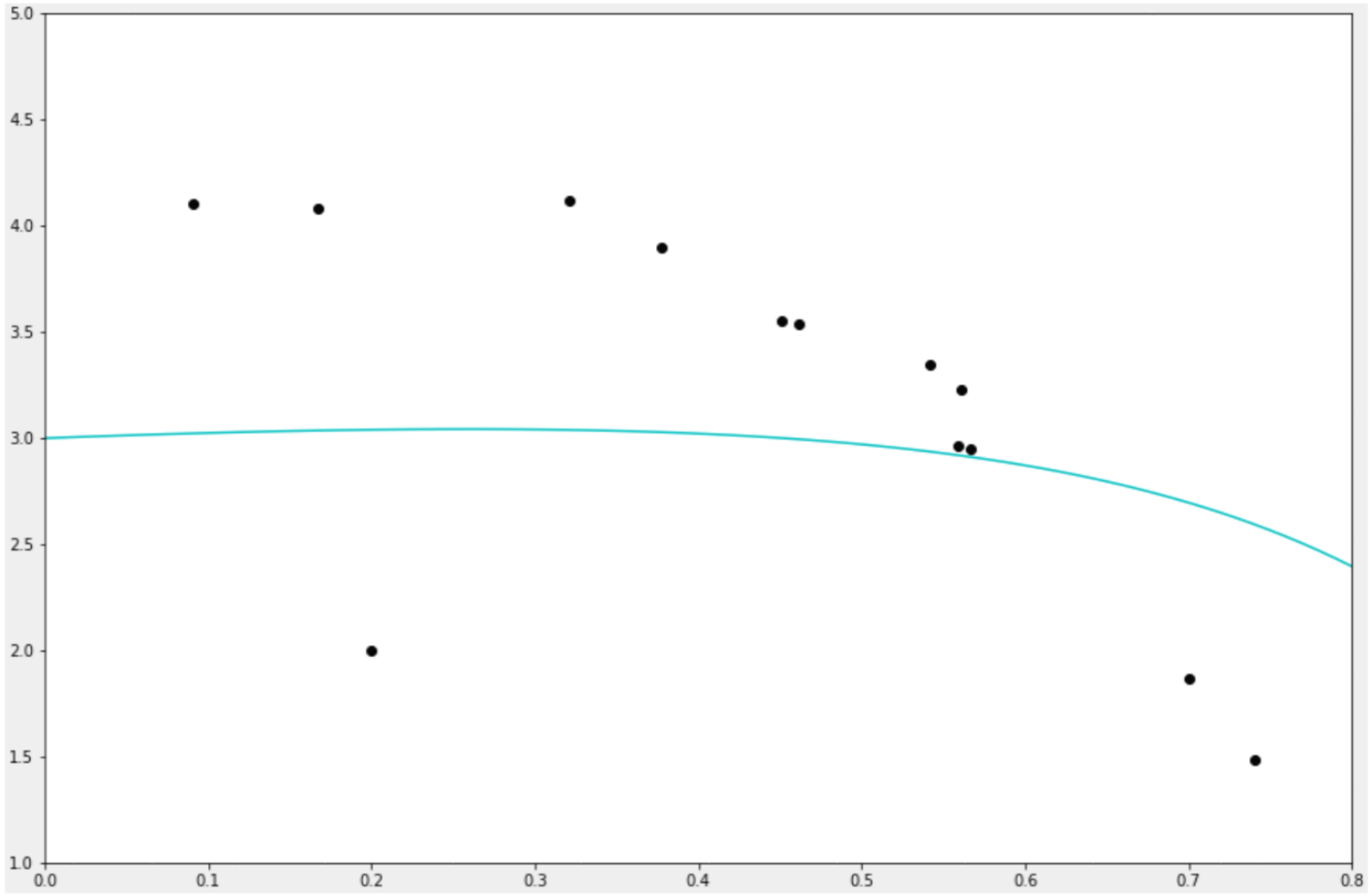

5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8


5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8



5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8


5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8


5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

Reducing the influence of high degree features will soften the curves of $h(x)$

Regularized Cost function and Gradient Descent

Linear and Logistic models regularize h(x) by applying cost penalties to h(x). It adds cost equal to theta values attached to features. This penalizes large theta values, thus encouraging smaller theta values

Linear Regression Regularized Cost

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)^2 + \frac{\boldsymbol{\lambda}}{\mathbf{2}\boldsymbol{m}}\sum_{j=1}^{n}\boldsymbol{\theta}_j^2$$

Linear Regression Regularized Gradient Descent

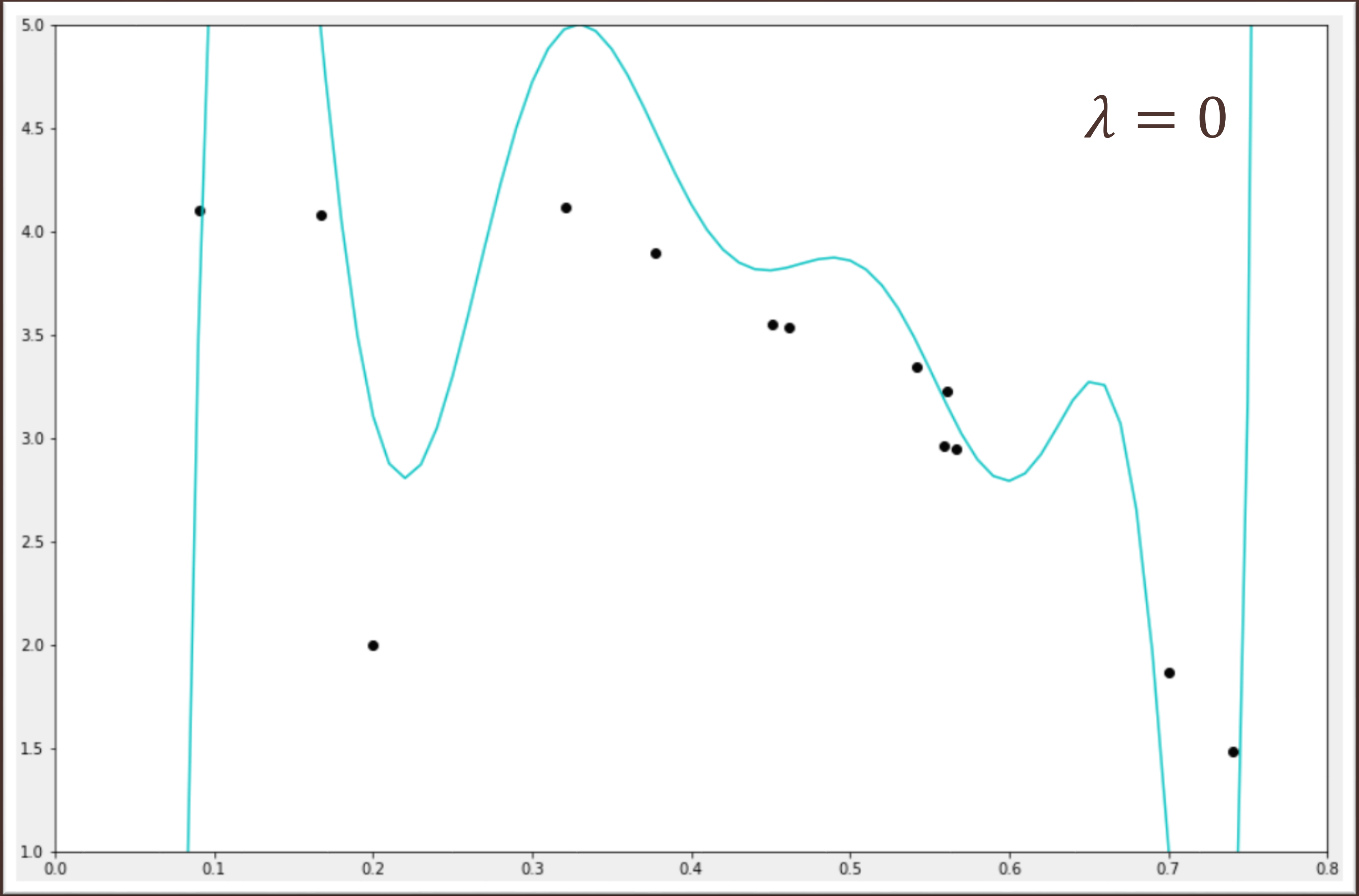
$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)} + \frac{\boldsymbol{\lambda}}{\boldsymbol{m}}\sum_{j=1}^{n}\boldsymbol{\theta}_j$$

Logistic Regression Regularized Cost

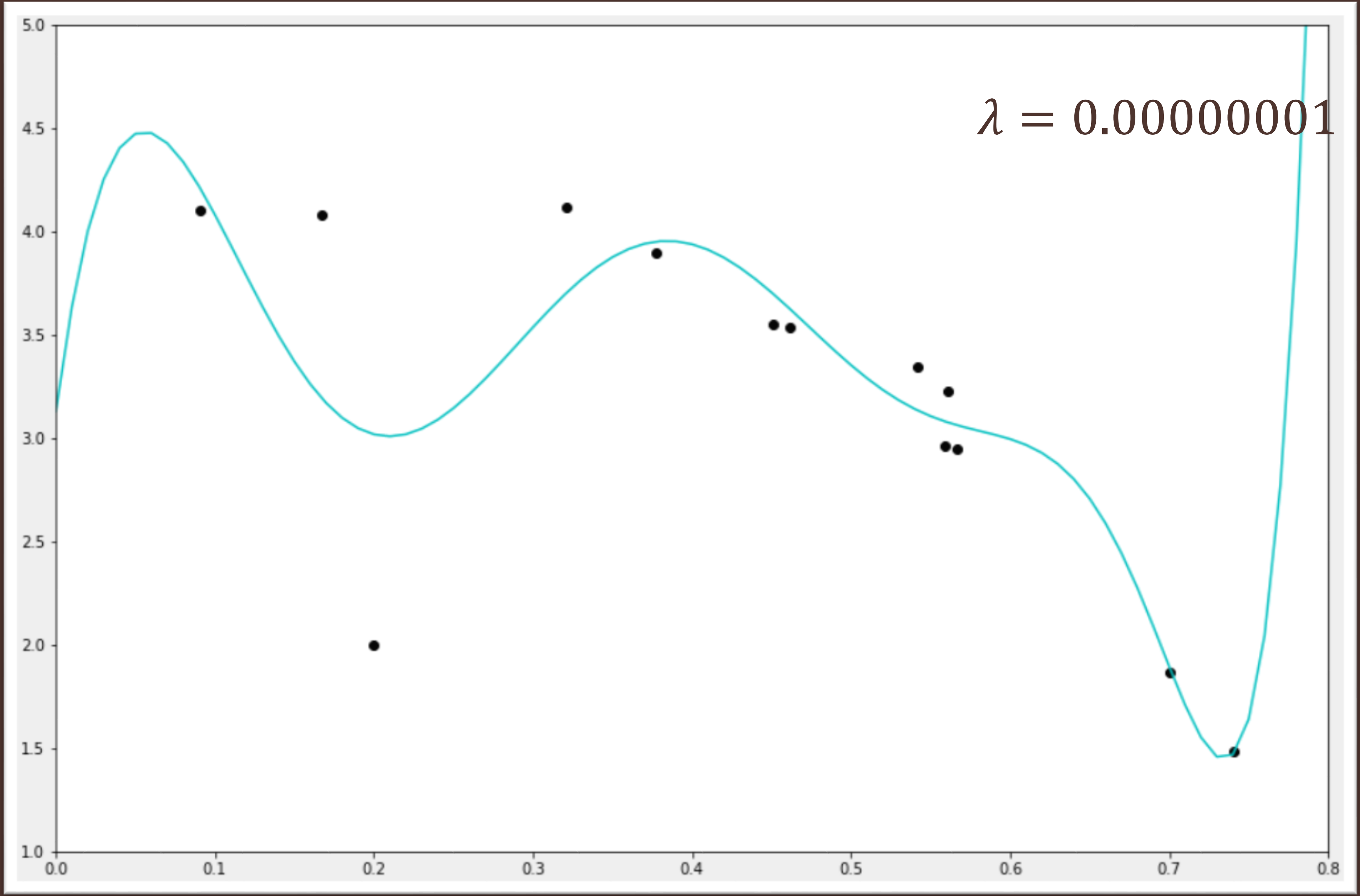
$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))] + \frac{\boldsymbol{\lambda}}{\mathbf{2}\boldsymbol{m}}\sum_{\boldsymbol{j}=\mathbf{1}}^{\boldsymbol{n}}\boldsymbol{\theta}_{\boldsymbol{j}}^{\mathbf{2}}$$

Logistic Regression Regularized Gradient Descent

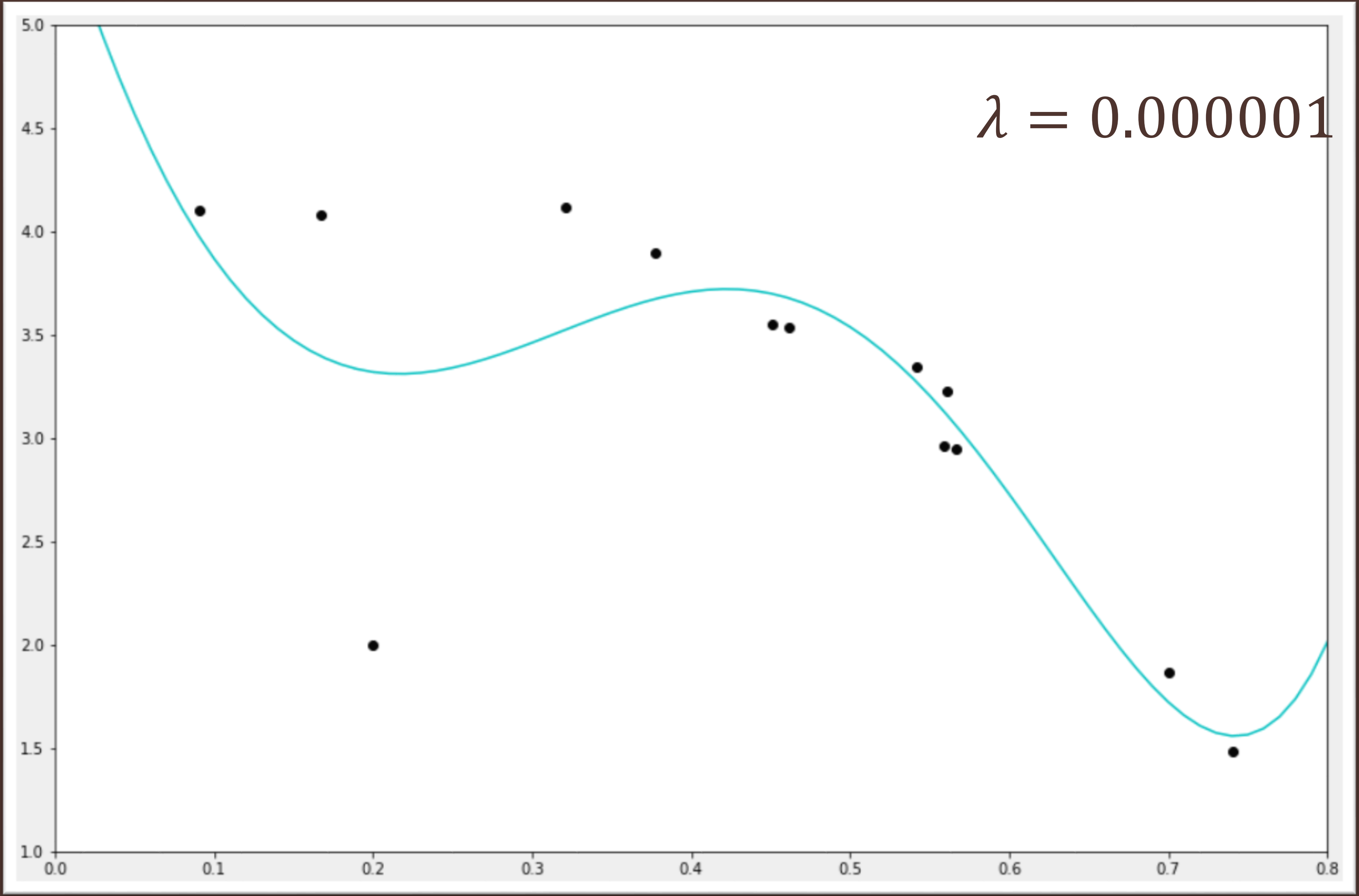
$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)} + \frac{\boldsymbol{\lambda}}{\boldsymbol{m}}\sum_{\boldsymbol{j}=\mathbf{1}}^{\boldsymbol{n}}\boldsymbol{\theta}_{\boldsymbol{j}}$$



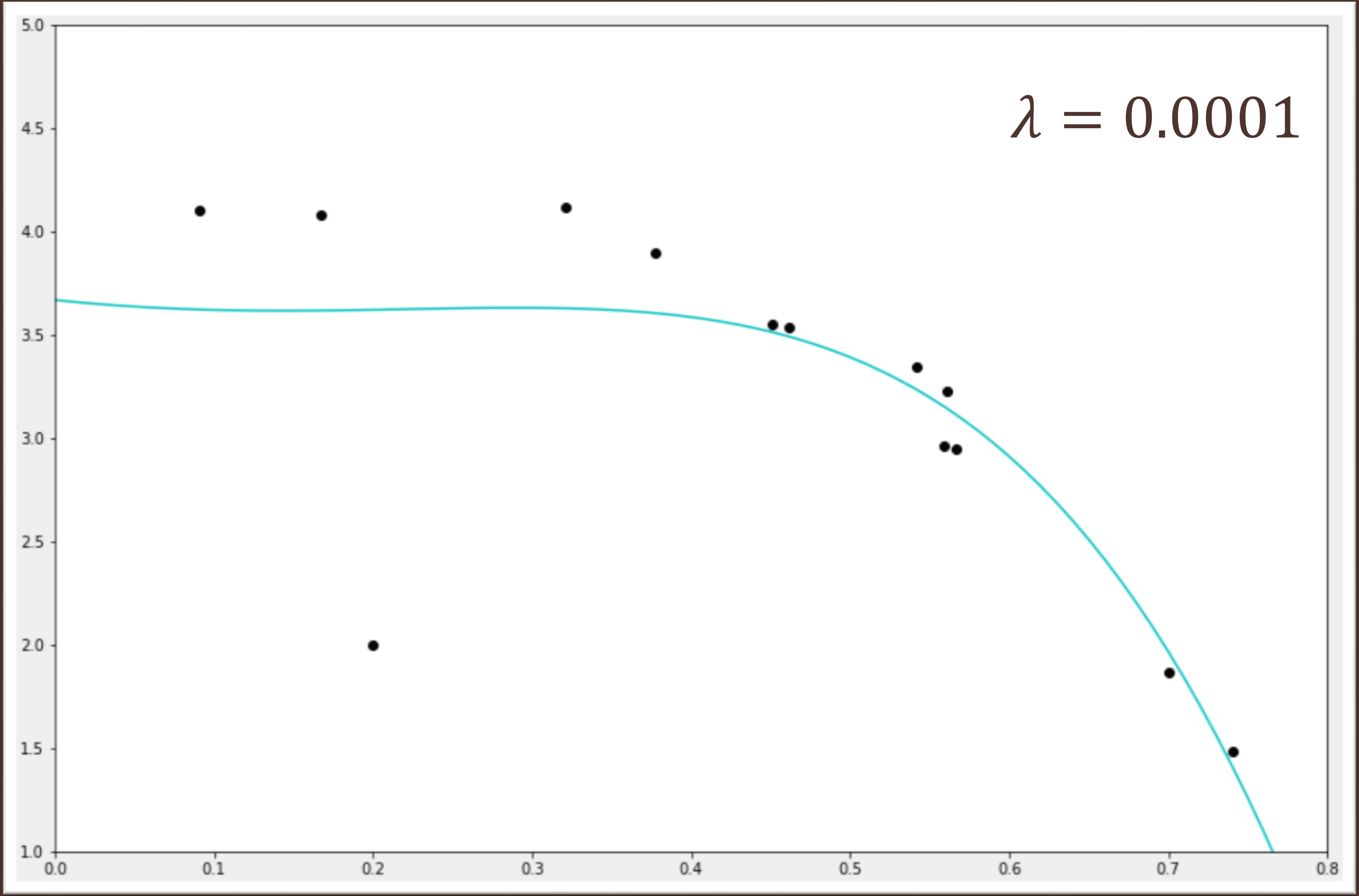
$\lambda = 0$
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8



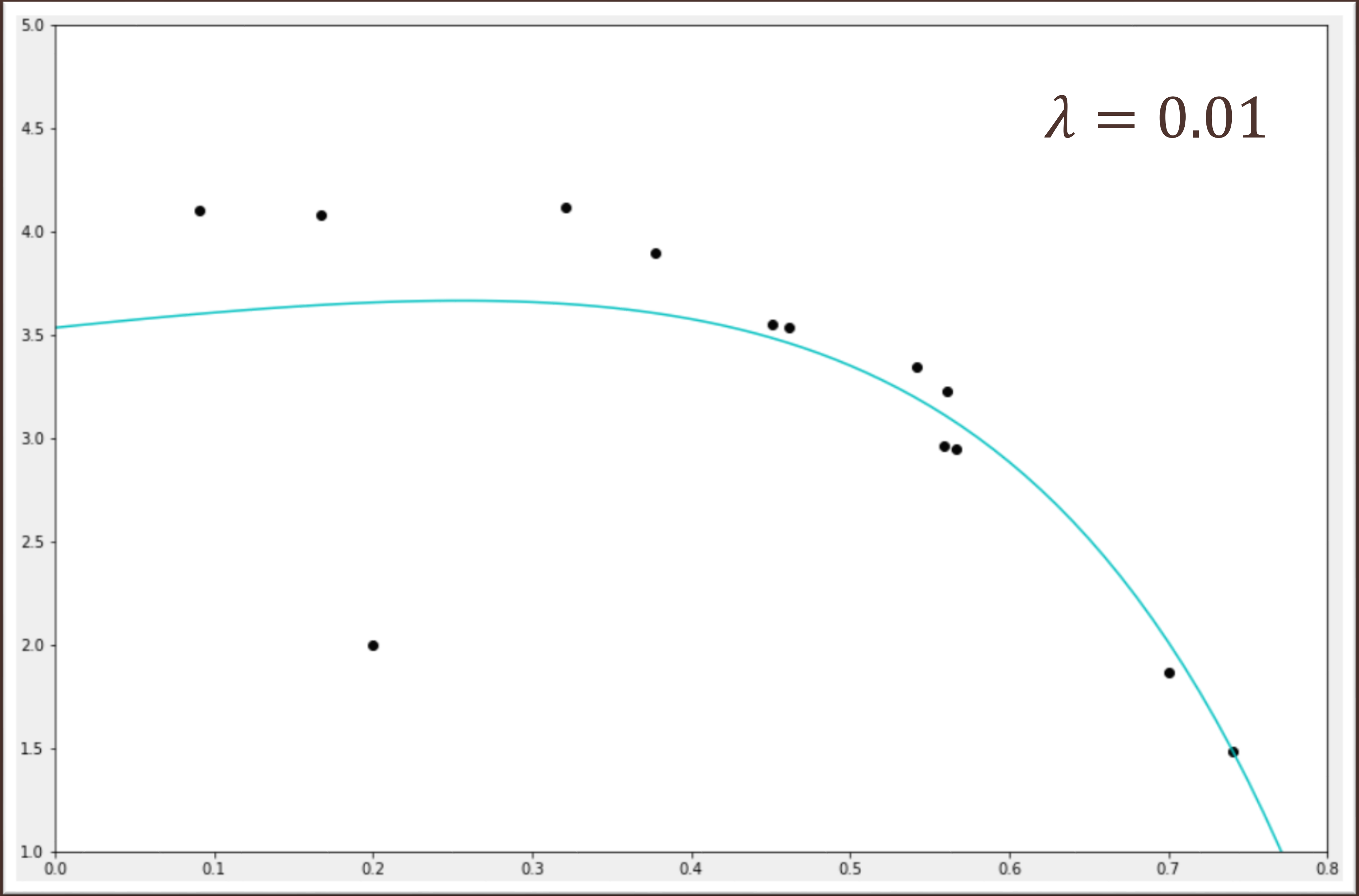
$\lambda = 0.00000001$



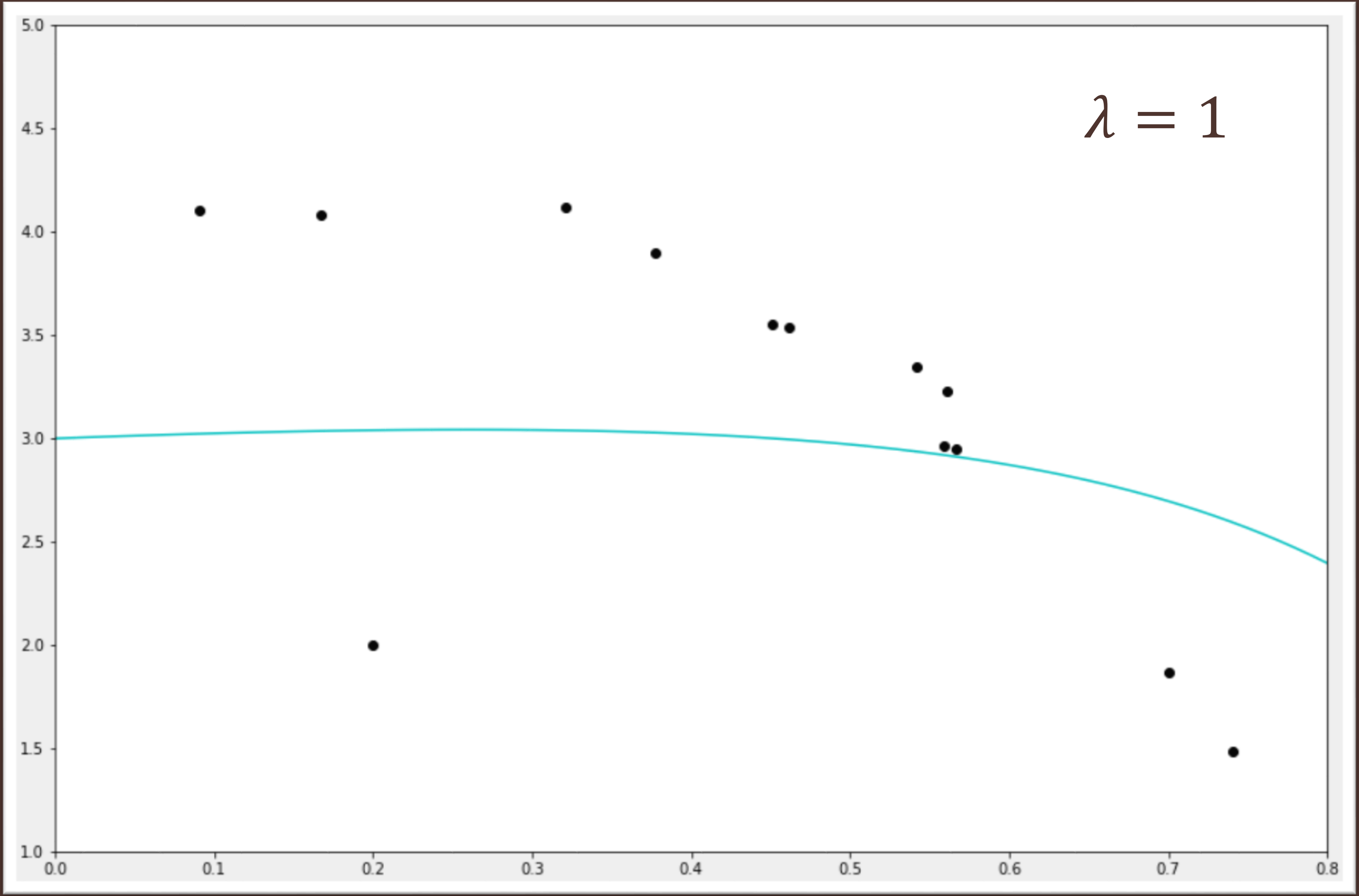
$\lambda = 0.000001$



$\lambda = 0.0001$



$\lambda = 0.01$



$\lambda = 1$