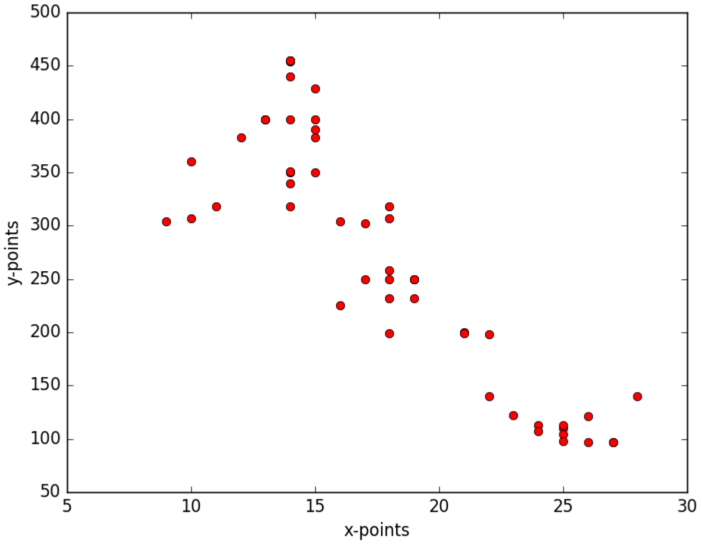
3.1 Linear Regression

a) see File *linear-regression.py* (marked in comments)

b) It is quite obvious that this output can be approximated by a function of degree one.



c) Equation for such a model:

$$\hat{y} = w * x + b$$

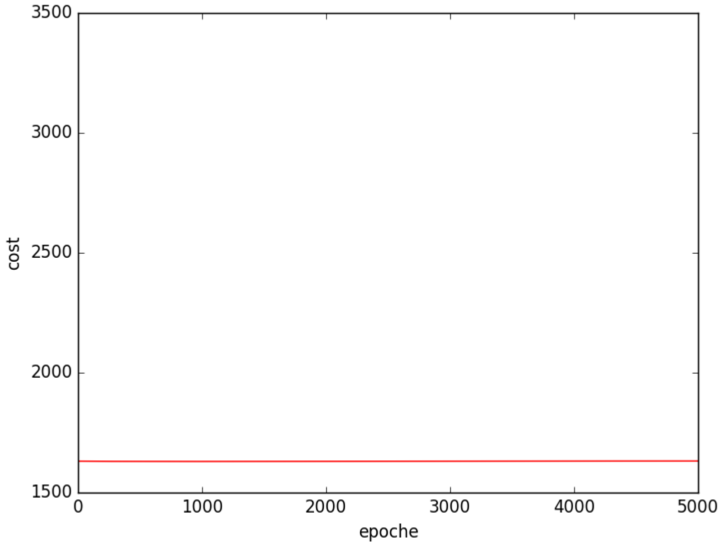
Gradient descent update rule:

$$\nabla_W \text{MSE}_{\text{train}} = \nabla_W \frac{1}{m} \|(X^{(\text{train})} * w - y^{(\text{train})})\|_2^2$$

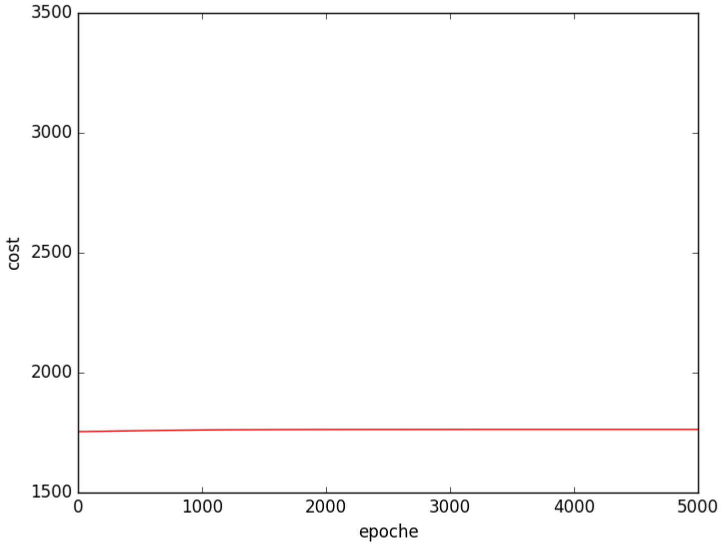
d)

- we are initiating random values for W and b and after multiple executions (using learning rate 0.01 and 5000 epochs) we noticed that W seems tend towards -20 and b towards 619. We decided to use these values as our initial values for W and b.
- After using these initial values and running the same procedure our gradient descent "resulted" in $W = -20.49593163$ and $b = 620.40246582$

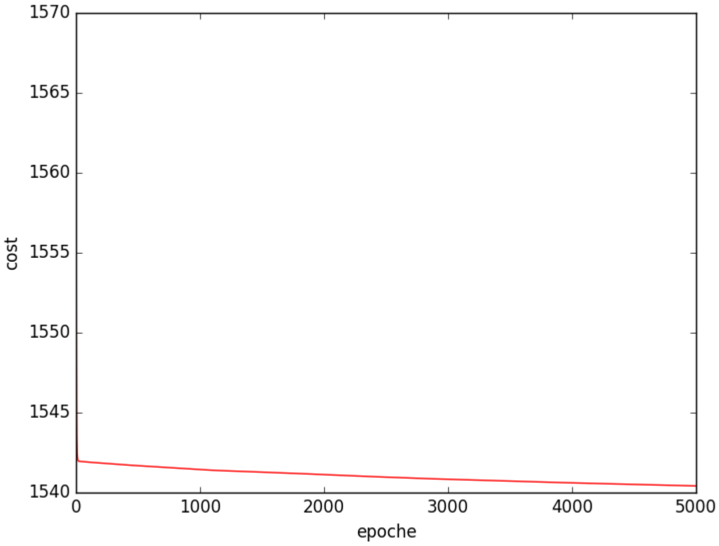
using the initial values above and *epochs* = 5000 we are capturing the final loss after using a specific learningrate:



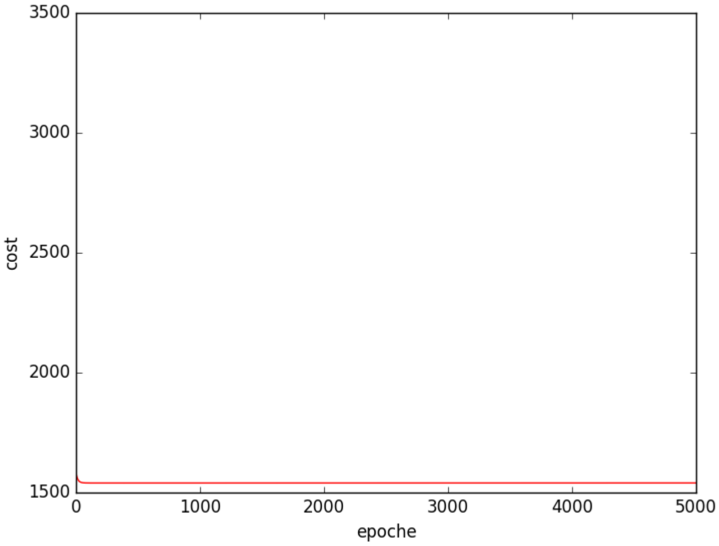
learningrate = 0.01 ⇒ Cost = 1631.553466797



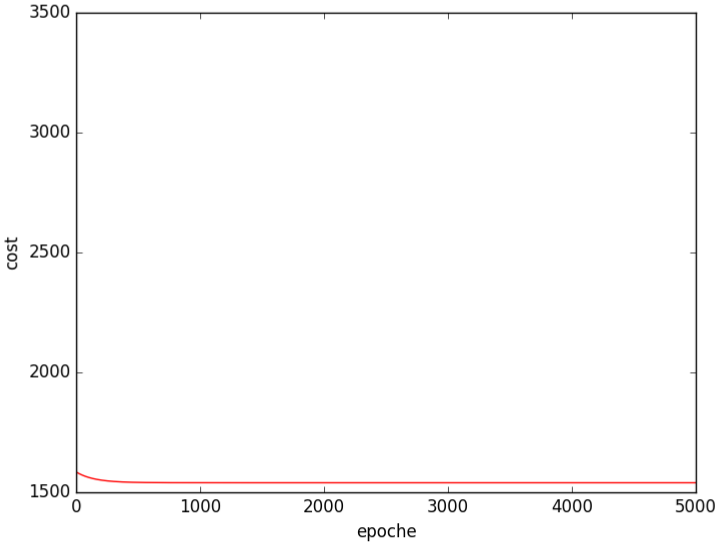
learningrate = 0.03 ⇒ Cost = 1763.359008789



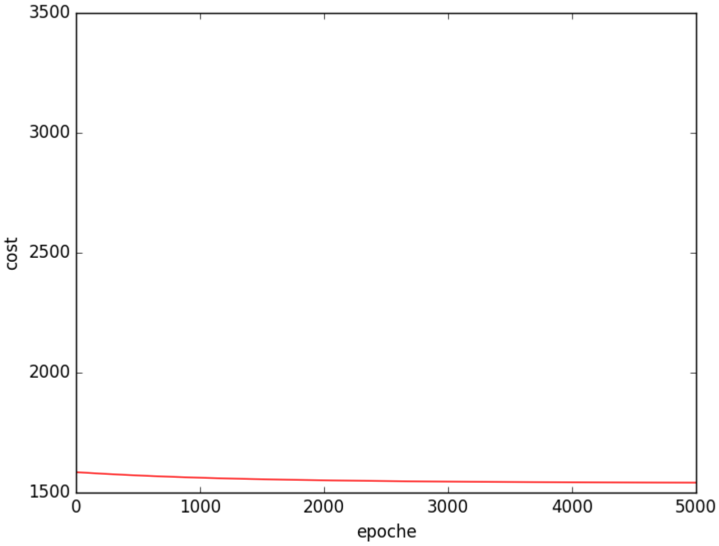
learningrate = 0.001 ⇒ Cost = 1540.411987305



learningrate = 0.0001 ⇒ Cost = 1539.946899414

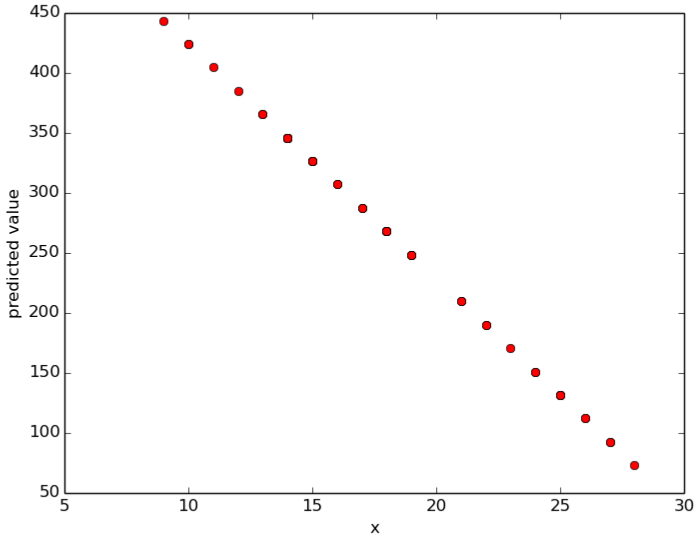


learningrate = 0.00001 ⇒ Cost = 1539.785766602



learningrate = 0.000001 ⇒ Cost = 1541.292480269

e) for this task we just use the best value from the task before (learningrate = 0.00001) which already converged to Cost = 1539.785766602 after 5000 epochs.



learningrate = 0.000001 ⇒ Predictions are shown in the picture

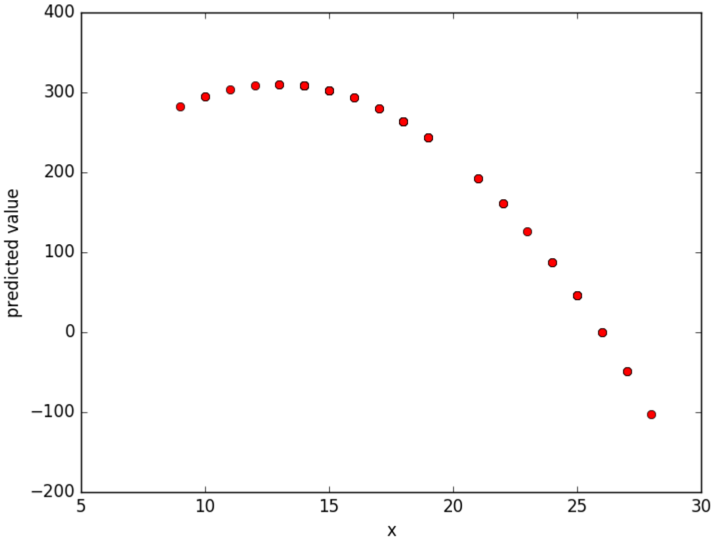
f) Equation for such a model:

$$\hat{y} = w_1 * x + w_2 * x^2 + b$$

Gradient descent update rule:

$$\nabla_W \text{MSE}_{\text{train}} = \nabla_W \frac{1}{m} \|(\hat{y} - y^{(\text{train})})\|_2^2 \text{ with } \hat{y} \text{ defined as above}$$

g) Result:



prediction using learning rate 0.0001 and 1800 epochs produced the best results with a cost of 3600.

h) no solution

i) no solution

3.2 Regularization

no solution