1) Linear Regression

Exercise 3.1

a, b)

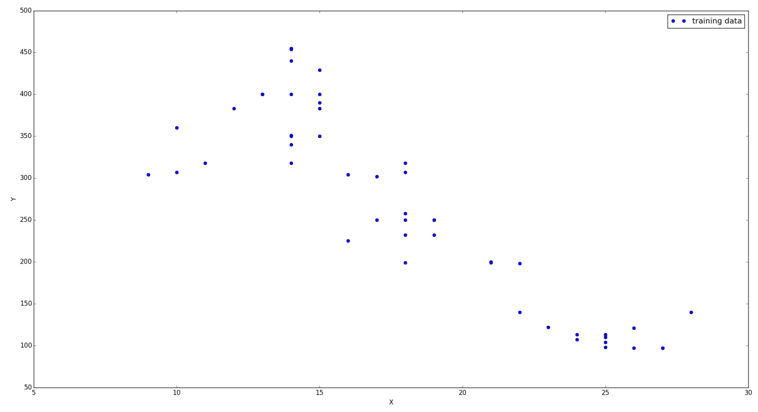

Figure 1: Training data points

From the figure 1, we think that the the polynomial function that estimate these data points has degree 3, because we can see that there are 2 stationary points: 1 local maximum at **x** around [12,15] and 1 local minimum at **x** around [25,27]

c)

With first order model, we have the equation of the polynomial model:

$$y = ax + b = Xw$$

Where X is the Vandermonde matrix of x of degree 1, meaning $X = \begin{pmatrix} x^1 & x^0 \end{pmatrix}$ and w is the weight vector with $w = \begin{pmatrix} a \\ b \end{pmatrix}$.
We use the mean square error as the loss function (we multiplied it by 0.5 so we don't have to multiply the gradient by 2.):

$$L = \frac{1}{2n}\sum_{i=1}^{n}(y_i - (ax_i + b))^2 = \frac{1}{n}(Xw - y)^T(Xw - y)$$

Hence, the first derivative with respect to parameter **a** and **b**:

$$\frac{\partial L}{\partial a} = \frac{1}{n}\sum_{i=1}^{n} -x_i(y_i - ax_i - b)$$
$$\frac{\partial L}{\partial b} = \frac{1}{n}\sum_{i=1}^{n} -(y_i - ax_i - b)$$

So, we have update rule:

$$\begin{aligned} a_{new} &= a_{old} - \alpha \frac{\partial L}{\partial a} \\ &= a_{old} + \frac{\alpha}{n}\sum_{i=1}^{n} x_i(y_i - ax_i - b) \\ b_{new} &= b_{old} - \alpha \frac{\partial L}{\partial b} \\ &= b_{old} + \frac{\alpha}{n}\sum_{i=1}^{n}(y_i - ax_i - b) \end{aligned}$$

Or in vectorized form:

$$\begin{aligned} \frac{\partial L}{\partial w} &= \frac{1}{n}X^T(Xw - y) \\ w_{new} &= w_{old} - \alpha \frac{\partial L}{\partial w} \\ &= w_{old} - \frac{\alpha}{n}X^T(Xw - y) \end{aligned}$$

Where n = 50, which is the number of training data points; and α is learning rate.

d, e)

We initialized the weights w randomly and the learning rate as $\alpha = 0.0028$
We stopped at 10000 iterations as the loss did not seem to change much.
The parameter we found were: $w = (a, b) = (-15.74831151, 546.24744732)$. The final mean squared error that we get is ≈ 3559.44. The final cost function value is ≈ 1779.72 as we half the MSE for numerical reasons.

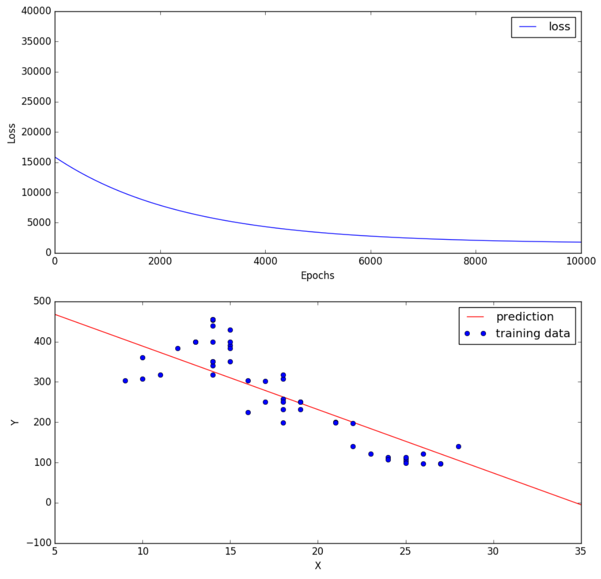

Figure 2: Loss function after each epoch of gradient descent (top) and learned first order model (bottom).

f)

With second order model, we have the equation of the polynomial model:

$$y = ax^2 + bx + c = Xw$$

Where X is the Vandermonde matrix of x of degree two, meaning $X = \begin{pmatrix} x^2 & x^1 & x^0 \end{pmatrix}$ and w is the weight vector $w = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$

We use the mean squared error multiplied by 0.5 as the loss function:

$$L = \frac{1}{2n}\sum_{i=1}^{n}(y_i - (ax_i^2 + bx_i + c))^2 = \frac{1}{2n}(Xw - y)^T(Xw - y)$$

Hence, the first derivatives with respect to parameters **a**, **b** and **c** are:

$$\begin{aligned}
\frac{\partial L}{\partial a} &= \frac{1}{n}\sum_{i=1}^{n} -x_i^2(y_i - ax_i^2 - bx_i - c) \\
\frac{\partial L}{\partial b} &= \frac{1}{n}\sum_{i=1}^{n} -x_i(y_i - ax_i^2 - bx_i - c) \\
\frac{\partial L}{\partial c} &= \frac{1}{n}\sum_{i=1}^{n} -(y_i - ax_i^2 - bx_i - c)
\end{aligned}$$

or

$$\frac{\partial L}{\partial w} = \frac{1}{n}X^T(Xw - y)$$

So, we have update rule:

$$\begin{aligned}
a_{new} &= a_{old} - \alpha\frac{\partial L}{\partial a} \\
&= a_{old} + \frac{\alpha}{n}\sum_{i=1}^{n} x_i^2(y_i - ax_i^2 - bx_i - c) \\
b_{new} &= b_{old} - \alpha\frac{\partial L}{\partial b} \\
&= b_{old} + \frac{\alpha}{n}\sum_{i=1}^{n} x_i(y_i - ax_i^2 - bx_i - c) \\
c_{new} &= c_{old} - \alpha\frac{\partial L}{\partial c} \\
&= c_{old} + \frac{\alpha}{n}\sum_{i=1}^{n}(y_i - ax_i^2 - bx_i - c)
\end{aligned}$$

Vectorized:

$$\begin{aligned}
w_{new} &= w_{old} - \alpha\frac{\partial L}{\partial w} \\
&= w_{old} - \frac{\alpha}{n}X^T(Xw - y)
\end{aligned}$$

Where n = 50, which is the number of training data points; and ϵ is learning rate.

g)

We use the learning rate $\alpha = 0.000012$ and limit the number of epochs to 120000 and below is the loss function after each epoch and the corresponding trained model:

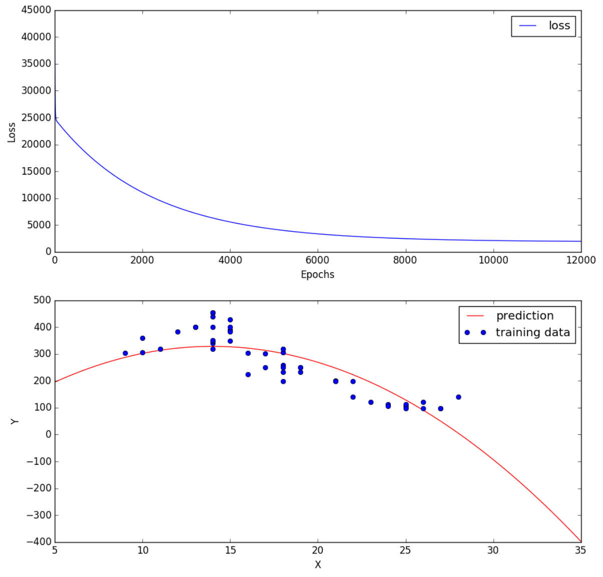

Figure 3: Loss function after each epoch in second order model (top) and learned second order model (bottom).

Parameters found are $w = (a, b, c) = (-1.64584803, 46.06328583, 6.13825731)$ with loss value ≈ 1971.82 and MSE ≈ 3943.64.

However, this is not expected value we need, since the algorithm will continue to optimize further than 120000 epochs. We implemented the Newton method to find the expected value (see **31g_newton.py**):

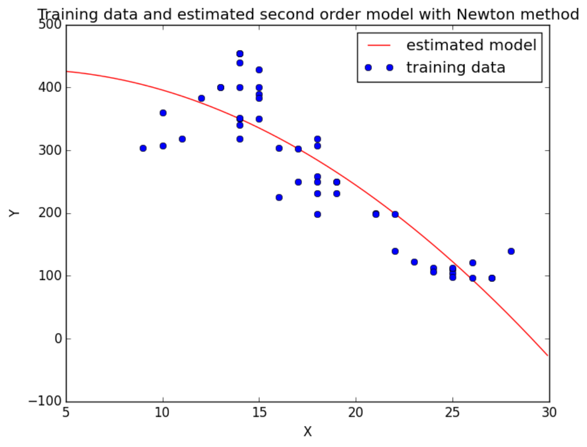

Figure 4: Learned second order model with gradient descent

The expected model is $w = (a, b, c) = (-0.613853733604, 3.249988125574, 424.868176858792)$ with MSE ≈ 2844.28 (see Figure 4)

h)

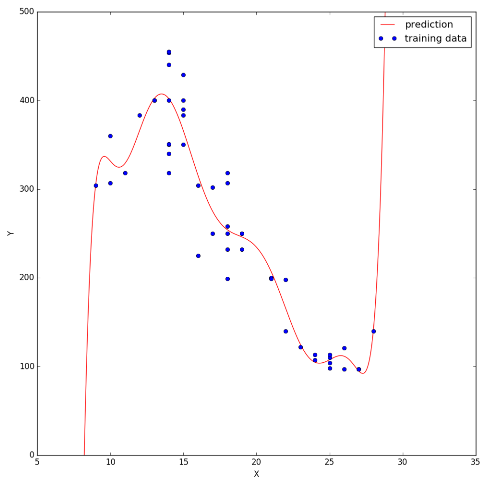

Figure 5: Training data and estimated 9th degree polynomial model

With this model, the MSE is decreased to ≈ 1136.02, which is better than first and second order models, however as the graph suggests, the new model looks way too overfitted.

i)

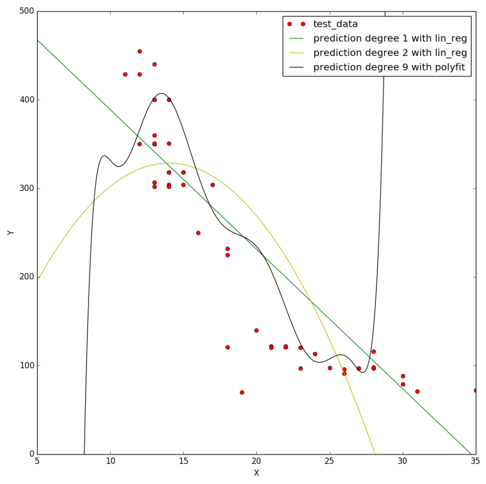

Figure 6: Test data and trained models

MSE first order model for the test set ≈ 3389.72
MSE second order model for test set ≈ 12562.16
MSE 9th order model for test set ≈ 1697820189.03

From the Figure 6 and the MSEs, we conclude that although the first model order fits worst for the training data, it fits best for the test data. In contrast, the ninth order model fits best for the training data, however it fits worst for the test data. Hence, the second and 9th order models are too overfitted to the training data and do not generalize well for unknown data points.

2) Regularization

Exercise 3.2

We can write the expression as:

$$J(\mathbf{w}) = MSE_{train} + \lambda w^T w = \|Xw - y\|^2 + \lambda w^T w = \|\tilde{X}w - \tilde{y}\|^2$$

Where X is a Vandermonde matrix of the data (given an arbitrary degree) and $\tilde{X} = \begin{pmatrix} X \\ \sqrt{\lambda} I \end{pmatrix}$.

y is the class label vector and $\tilde{y} = \begin{pmatrix} y \\ 0 \end{pmatrix}$ and w is the weight vector. Since we want to minimize the expression $\|\tilde{X}w - \tilde{y}\|^2$, we equate it to 0 and solve for w to get our closed form expression:

$$\begin{aligned}
\|\tilde{X}w - \tilde{y}\|^2 = (\tilde{X}w - \tilde{y})^T(\tilde{X}w - \tilde{y}) &= 0 \\
w^T \tilde{X}^T \tilde{X} w - 2 w^T \tilde{X} \tilde{y} + \tilde{y}^T \tilde{y} &= 0 \\
2\tilde{X}^T \tilde{X} w - 2\tilde{X}^T \tilde{y} &= 0 \\
w &= (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T \tilde{y} \\
w &= (X^T X + \lambda I)^{-1} X^T y
\end{aligned}$$