

# Linear Regression

## Linear Regression with One Variable

Linear regression is a supervised-learning algorithm which aims to model the linear relationship between the continuous dependent variable and independent variable.

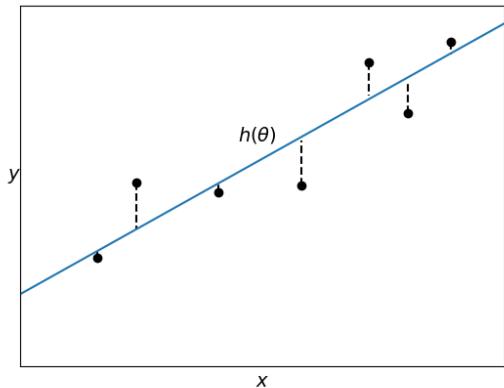


Figure 1: The line  $h_{\theta}(x)$  represents a possible hypothesis function of the linear regression model for this two-dimensional dataset. The dotted lines between the hypothesis function and the samples represent the differences between the actual values (y-coordinate) and the values predicted by the hypothesis function. This can be interpreted as the error in the prediction of a sample value.

The univariate linear regression hypothesis function is

$$h_{\theta}(x) = \theta_0 + \theta_1 x \quad (1)$$

This can be represented as a line in the xy-plane which intercepts the y-axis at  $\theta_0$  and has the gradient  $\theta_1$ .

The goal in linear regression is to choose the hypothesis function that produces the best fit to the data. This will allow us to make accurate predictions on future samples. In Figure 1 we can see that there are an infinite number of lines with different values of the parameters  $\theta_0$  and  $\theta_1$  that can be fit to the data, so how do we identify the one that produces the best fit?

We begin with introducing the notation  $x^i$  which denotes the feature of the  $i^{th}$  training sample. Likewise  $y^i$  denotes the  $y$  value of the  $i^{th}$  training sample.

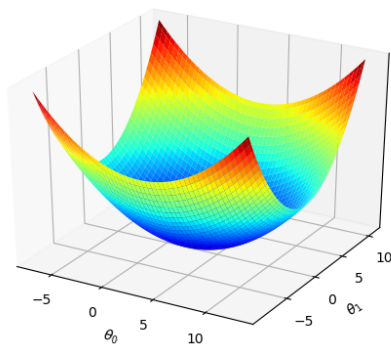
The error between the actual  $y$  value of a sample and the predicted value of the sample,  $h_{\theta}(x^i)$ , is given by  $h_{\theta}(x^i) - y^i$ .

To find the optimal parameters  $\theta_0$  and  $\theta_1$  we introduce the cost function

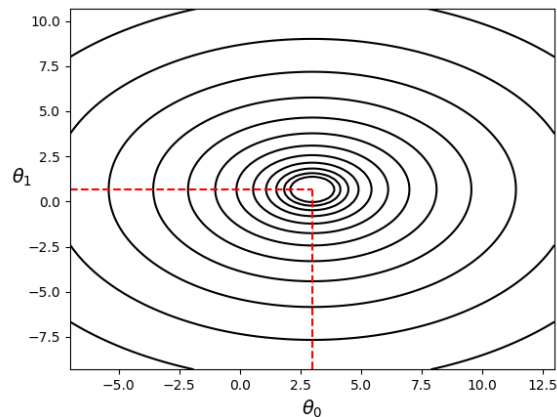
$$C(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m [h_{\theta}(x^i) - y^i]^2 \quad (2)$$

This is the sum of the squared errors for all  $m$  samples, divided by twice the number of samples, and is equal to the mean squared error (MSE) multiplied by a factor of 1/2 (we will see why we have this factor later).

The optimal regression line for the data will be the one that produces the smallest errors between the  $h_{\theta}(x^i)$  and the  $y^i$ . We can clearly see that this corresponds to minimising the cost function  $C(\theta_0, \theta_1)$ .



(a) Cost function surface plot.



(b) Contour plot of the same cost function.

Figure 2: (a) and (b) represent visualisations of the same cost function for some dataset in three and two dimensions, respectively. The minimum of the cost function is located where the dotted lines meet in (b).

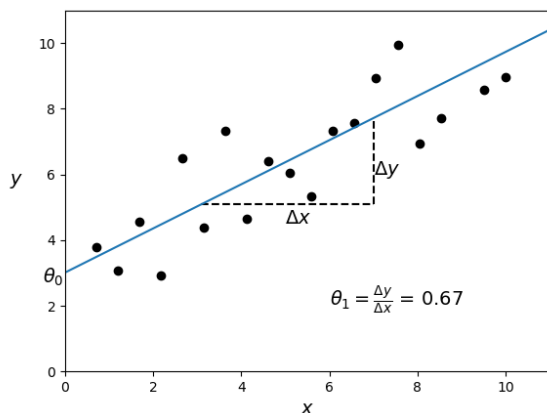


Figure 3: The linear regression model associated with the cost function depicted in Figure 2. Note that the parameters of  $\theta_0$  and  $\theta_1$  required for the best fit of the hypothesis function (represented by the blue line) to the data correspond to the cost function minimum (Figure 2(b)).

As a function of two variables,  $\theta_0$  and  $\theta_1$ , the cost function represents a two-dimensional surface. It can be shown (see Appendix) that the MSE is always convex in linear regression, and therefore has a unique global minimum. The cost function, being just the MSE multiplied by a factor, also has these properties. The values of  $\theta_0$  and  $\theta_1$  at this minimum are the values that produce the hypothesis function with the best fit to the data.

## Gradient Descent

So far we have determined that the values of  $\theta_0$  and  $\theta_1$  that result in an optimal fit of the hypothesis function to the data correspond to the location of the cost function minimum. One of the most common methods for minimising the cost function is gradient descent; the idea being to descend the convex surface of the cost function until the minimum is reached. Repeated steps are taken in the direction opposite to the gradient at the current point, which is the direction of steepest descent.

The gradient vector is given by

$$\nabla C = \begin{pmatrix} \frac{\partial C}{\partial \theta_0} \\ \frac{\partial C}{\partial \theta_1} \end{pmatrix} \quad (3)$$

Starting at the value  $\theta_j$  we apply the step

$$\theta_j \rightarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} [C(\theta_0, \theta_1)], \quad j = 0, 1 \quad (4)$$

where  $\alpha$  is known as the learning rate and determines the size of the step taken. To descend the cost function in the direction opposite to that of the gradient, this step must be implemented simultaneously for  $\theta_0$  and  $\theta_1$ .

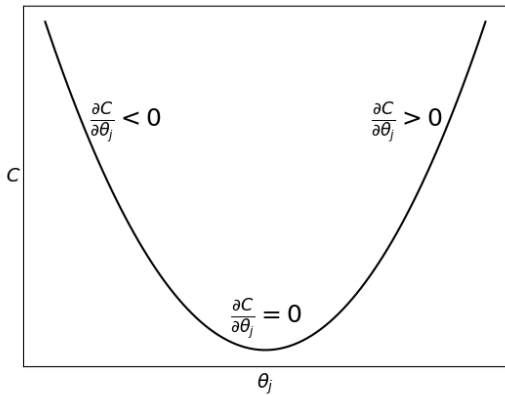


Figure 4: For values of  $\theta_j$  smaller than the value at the minimum, the rate of change of the cost function with respect to  $\theta_j$  is negative. For values of  $\theta_j$  larger than the value at the minimum, the rate of change is positive, and at the minimum it is equal to zero.

At  $\theta_j$ , we subtract the partial derivative  $\partial C / \partial \theta_j$  (multiplied by  $\alpha$ ). If the minimum occurs at a value smaller than  $\theta_j$  then we know from the convex shape of the cost function that  $\partial C / \partial \theta_j > 0$  (Figure 4). From equation (4) we can see that this will result in  $\theta_j$  being reassigned to a smaller value ready for the next iterative step. Conversely, if the minimum occurs at a value larger than  $\theta_j$ , then  $\partial C / \partial \theta_j < 0$ , and  $\theta_j$  will be reassigned to a larger value. Finally, when we are at the minimum,  $\partial C / \partial \theta_j = 0$ , and no changes are made to  $\theta_j$ , i.e., gradient descent stops.

Now let's compute the partial derivatives of  $C$ :

$$\frac{\partial C}{\partial \theta_0} = \frac{1}{2m} \frac{\partial}{\partial \theta_0} \sum_{i=1}^m (\theta_0 + \theta_1 x^i - y^i)^2 = \frac{2}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 x^i - y^i) = \frac{1}{m} \sum_{i=1}^m [h_{\theta}(x^i) - y^i] \quad (5)$$

$$\frac{\partial C}{\partial \theta_1} = \frac{1}{2m} \frac{\partial}{\partial \theta_1} \sum_{i=1}^m (\theta_0 + \theta_1 x^i - y^i)^2 = \frac{2}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 x^i - y^i) x^i = \frac{1}{m} \sum_{i=1}^m [h_\theta(x^i) - y^i] x^i \quad (6)$$

It becomes apparent why the MSE was multiplied by 1/2 for the cost function; it cancels out the factor of 2 obtained in the differentiation.

Using equations (4), (5), and (6) the gradient descent algorithm becomes

$$\theta_0 \rightarrow \theta_0 - \frac{\alpha}{m} \sum_{i=1}^m [h_\theta(x^i) - y^i], \quad \theta_1 \rightarrow \theta_1 - \frac{\alpha}{m} \sum_{i=1}^m [h_\theta(x^i) - y^i] x^i \quad (7)$$

## Multivariable Linear Regression

If we want to model the linear relationship between a dependent variable and more than one independent variables we employ the hypothesis function

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots + \theta_n x_n \quad (8)$$

If we define  $x_0 = 1$ , then we can write

$$h_\theta(x) = \sum_{i=0}^n \theta_i x_i = \Theta^T \mathbf{X} \quad (9)$$

where

$$\Theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \quad (10)$$

This is the equation of an n-dimensional hyperplane.

The multivariable linear regression cost function takes the same form as in the single variable case

$$C(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^m [h_\theta(x^i) - y^i]^2 \quad (11)$$

To find the best hypothesis for the data we again look to minimise the errors between the  $h_\theta(x^i)$  and the  $y^i$ , which is equivalent to minimising the cost function. The values of the  $\theta_i$  obtained will define the hyperplane that is the optimal hypothesis function.

The partial derivatives of  $C$  are

$$\frac{\partial C}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^m [h_\theta(x^i) - y^i] \quad (12)$$

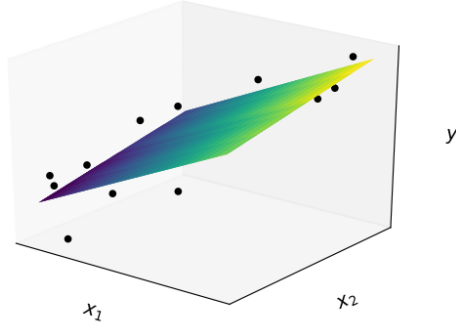


Figure 5: Here we have a dataset with two independent variables,  $x_1$  and  $x_2$ . The hypothesis function forms a plane which is chosen to minimise the squared vertical distances between the samples and the plane.

and

$$\frac{\partial C}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^m [h_{\theta}(x^i) - y^i] x_j^i, \quad \text{for } j = 1, 2, \dots, n \quad (13)$$

The gradient descent algorithm is

$$\theta_0 \rightarrow \theta_0 - \frac{\alpha}{m} \sum_{i=1}^m [h_{\theta}(x^i) - y^i], \quad \theta_j \rightarrow \theta_j - \frac{\alpha}{m} \sum_{i=1}^m [h_{\theta}(x^i) - y^i] x_j^i, \quad \text{for } j = 1, 2, \dots, n \quad (14)$$

As in the single variable case, these steps must be performed simultaneously for all  $\theta_i$ , until the minimum is reached.

## The Normal Equation

The normal equation is an analytic approach to solving the cost function minimisation problem.

We start by writing a hypothesis function column vector containing the hypothesis function of each sample. If we have  $m$  samples with  $n$  features we can create the matrix  $X$  with dimensions  $(m \times n)$ , such that the hypothesis function column vector is equal to this matrix

multiplied by the column vector  $\Theta$ :

$$\mathbf{h}_\Theta(x) = \begin{pmatrix} h_\theta(x^1) \\ h_\theta(x^2) \\ \vdots \\ h_\theta(x^m) \end{pmatrix} \quad (15)$$

$$= \begin{pmatrix} \theta_0 + \theta_1 x_1^1 + \theta_2 x_2^1 + \theta_3 x_3^1 + \dots + \theta_n x_n^1 \\ \theta_0 + \theta_1 x_1^2 + \theta_2 x_2^2 + \theta_3 x_3^2 + \dots + \theta_n x_n^2 \\ \vdots \\ \theta_0 + \theta_1 x_1^m + \theta_2 x_2^m + \theta_3 x_3^m + \dots + \theta_n x_n^m \end{pmatrix} \quad (16)$$

$$= \begin{pmatrix} \leftarrow \mathbf{x}^1 \rightarrow \\ \leftarrow \mathbf{x}^2 \rightarrow \\ \vdots \\ \leftarrow \mathbf{x}^m \rightarrow \end{pmatrix} \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{pmatrix} = X\Theta \quad (17)$$

where  $\mathbf{x}^i$  is the vector representing the  $i^{th}$  sample, and  $x_j^i$  denotes the  $j^{th}$  feature of the  $i^{th}$  sample.

If we create a column vector of the y values of the samples

$$\mathbf{Y} = \begin{pmatrix} y^1 \\ y^2 \\ \vdots \\ y^m \end{pmatrix} \quad (18)$$

then

$$X\Theta - \mathbf{Y} = \begin{pmatrix} h_\theta(x^1) - y^1 \\ h_\theta(x^2) - y^2 \\ \vdots \\ h_\theta(x^m) - y^m \end{pmatrix} \quad (19)$$

Multiplying this vector by its transpose yields

$$\begin{aligned} (X\Theta - \mathbf{Y})^T (X\Theta - \mathbf{Y}) &= [h_\theta(x^1) - y^1]^2 + [h_\theta(x^2) - y^2]^2 + \dots + [h_\theta(x^m) - y^m]^2 \\ &= \sum_{i=1}^m [h_\theta(x^i) - y^i]^2 \end{aligned}$$

Therefore, the cost function can be expressed as

$$\begin{aligned} C(\Theta) &= \frac{1}{2m} (X\Theta - \mathbf{Y})^T (X\Theta - \mathbf{Y}) \\ &= \frac{1}{2m} [(X\Theta)^T (X\Theta) - (X\Theta)^T \mathbf{Y} - \mathbf{Y}^T (X\Theta) + \mathbf{Y}^T \mathbf{Y}] \\ &= \frac{1}{2m} [(X\Theta)^T X\Theta - 2(X\Theta)^T \mathbf{Y} + \mathbf{Y}^T \mathbf{Y}] \end{aligned} \quad (20)$$

where in the last line we have used the property that the dot product of vectors is symmetric with respect to its arguments, i.e.,  $(X\Theta)^T Y = Y^T (X\Theta)$ .

The goal is to minimise  $C$  and determine  $\Theta$ . So, first we differentiate the cost function with respect to  $\Theta$ :

$$\frac{dC}{d\Theta} = \frac{1}{2m}(2X^T X\Theta - 2X^T Y) \quad (21)$$

Next, we set the result equal to zero:

$$\begin{aligned} 0 &= \frac{1}{2m}(2X^T X\Theta - 2X^T Y) \\ X^T X\Theta &= X^T Y \end{aligned} \quad (22)$$

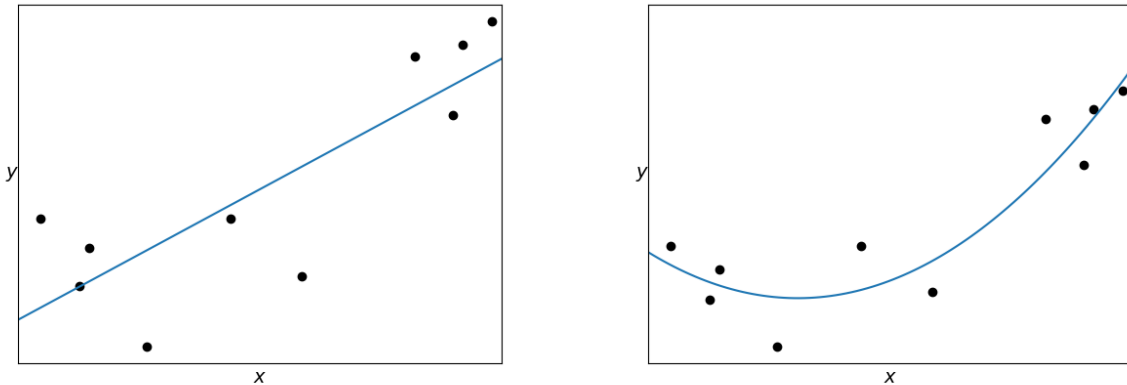
Assuming that there exists a matrix  $(X^T X)^{-1}$  such that  $(X^T X)^{-1}(X^T X) = I$ , i.e., that  $X^T X$  is invertible, we obtain the result

$$\Theta = (X^T X)^{-1}(X^T Y) \quad (23)$$

This is the normal equation, and it can be employed directly to find the  $\theta_i$ . When the data has few features, using the normal equation is often preferable to gradient descent. However, for large numbers of features, gradient descent is usually the better option. This is because  $X^T X$  has the dimensions  $(n \times n)$ , meaning it becomes more demanding to compute as  $n$  increases.

## Polynomial Regression

Linear regression can be employed in settings in which the relationship between the dependent variable and the independent variable is non-linear.



(a) Linear regression.

(b) Polynomial regression.

Figure 6: (a) The linear hypothesis function does not provide a good fit to this dataset, which is more suited to a quadratic hypothesis function (b).

The idea of polynomial regression is to use powers of the independent variable  $x$ , as additional features in the model. Therefore, the polynomial regression hypothesis function is given by

$$h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \dots + \theta_n x^n \quad (24)$$

Clearly, polynomial regression is a special case of multivariable linear regression.

In Figure 6(a) the linear hypothesis function  $h_{\theta}(x) = \theta_0 + \theta_1 x$  does not fit the data well. In Figure 6(b) we have used the quadratic hypothesis function  $h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2$  which results in an improved fit.

## Regularisation

Overfitting is a problem commonly encountered in Machine Learning. It occurs when a model generates an excellent fit to the training data but generalises poorly to the test data. The model learns noise that is not an inherent characteristic of the training data, to the extent that model performance on new data is adversely affected.

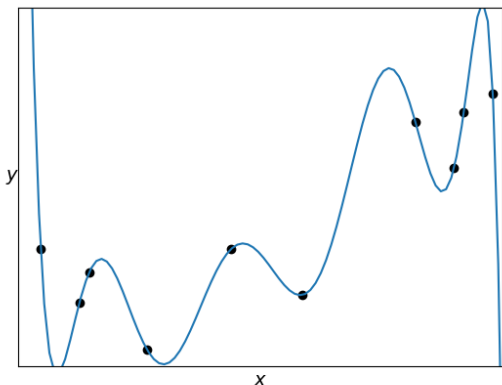


Figure 7: Here we have fitted a polynomial of degree 10 to the data from Figure 6. The hypothesis function passes through all samples, making it a perfect fit, but will the model generalise well to new data?

In Figure 7 we can see that the hypothesis function, a polynomial of degree 10, fits the training data perfectly. However, compared to the quadratic hypothesis function in Figure 6(b), the performance of the model on new data will be worse, especially at the extremes of the plot where the hypothesis tends to infinities. The problem is that the hypothesis function we are trying to fit is too complex for the data that we have, and as a result noise that is not a fundamental property of the data is learnt. So how do we go about rectifying this issue? In the present case where we have 10 features, we can reproduce the quadratic hypothesis function if we set the coefficients  $\theta_3, \theta_4, \dots, \theta_{10}$  equal to zero. This is the idea behind regularisation. We add a regularisation term to the cost function that aims to set all the  $\theta_i$  equal to zero as the cost function is minimised. The only  $\theta_i$  that remain large are those that are important in the minimisation of the original term in the cost function, i.e., those that ensure the greatest error reductions between the  $h_{\theta}(x^i)$  and the  $y^i$ .



The (ridge) regularisation term added to the cost function is  $\frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$ , where  $\lambda$  is the regularisation parameter that determines the severity of the penalty applied to the  $\theta_j$ . So, our regularised cost function for multivariable linear regression becomes

$$C(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \left\{ \sum_{i=1}^m [h_{\theta}(x^i) - y^i]^2 + \sum_{j=1}^n \theta_j^2 \right\} \quad (25)$$

If  $\lambda = 0$ , there is no regularisation. For increasingly larger values of  $\lambda$ , the parameters  $\theta_1, \theta_2, \dots, \theta_n$  will be penalised to a greater extent, i.e., they will tend to increasingly small values. If  $\lambda$  is very large then  $h_{\theta}(x) \rightarrow \theta_0$ , i.e., the hypothesis function will tend to a straight line and the data will be underfitted. This occurs when the hypothesis function is too simple to model the data.

The first term in the cost function aims to find the parameters that produce the best fit between the hypothesis function and the data, with no restriction on the values that the parameters can take or the complexity of the model. The sole goal of the regularisation term is to minimise the parameters, and produce a hypothesis function approximating a straight line. Therefore, the regularised cost function represents a balancing act between model complexity and simplicity. Large parameter values are only tolerated by the regularisation term if they make significant contributions to the reduction of the first term when the cost function is minimised.

$\partial C / \partial \theta_0$  remains unchanged with the addition of the regularisation term as the index  $j$  begins at 1. However,  $\partial C / \partial \theta_j$  must be computed again:

$$\frac{\partial C}{\partial \theta_j} = \frac{1}{m} \left\{ \sum_{i=1}^m [h_{\theta}(x^i) - y^i] x_j^i + \lambda \theta_j \right\}, \quad \text{for } j = 1, 2, \dots, n \quad (26)$$

As a consequence the gradient descent algorithm becomes

$$\theta_0 \rightarrow \theta_0 - \frac{\alpha}{m} \sum_{i=1}^m [h_{\theta}(x^i) - y^i], \quad \theta_j \rightarrow \theta_j - \frac{\alpha}{m} \left\{ \sum_{i=1}^m [h_{\theta}(x^i) - y^i] x_j^i + \lambda \theta_j \right\}, \quad \text{for } j = 1, 2, \dots, n \quad (27)$$

where as usual, the steps are repeated simultaneously for all  $\theta_i$ , until the minimum is reached.

## References

Ng, A. *Machine Learning*. Offered by Stanford University on Coursera.

## Appendix

### The MSE is always convex for linear regression:

The MSE for univariate linear regression is

$$MSE = \frac{1}{m} \sum_{i=1}^m (\theta_0 + \theta_1 x^i - y^i)^2 \quad (28)$$

Let's discard the  $1/m$  factor and define  $J^i(\theta_0, \theta_1)$  as the  $i^{th}$  term of the sum so that

$$J^i(\theta_0, \theta_1) = (\theta_0 + \theta_1 x^i - y^i)^2 \quad (29)$$

Now we find the second derivatives of  $J^i$ :

$$\frac{\partial^2 J^i}{\partial \theta_0^2} = 2, \quad \frac{\partial^2 J^i}{\partial \theta_1^2} = 2x^{i2}, \quad \frac{\partial^2 J^i}{\partial \theta_0 \partial \theta_1} = 2x^i, \quad \frac{\partial^2 J^i}{\partial \theta_1 \partial \theta_0} = 2x^i \quad (30)$$

The Hessian matrix of  $J^i$  is

$$H = \begin{pmatrix} \frac{\partial^2 J^i}{\partial \theta_0^2} & \frac{\partial^2 J^i}{\partial \theta_0 \partial \theta_1} \\ \frac{\partial^2 J^i}{\partial \theta_1 \partial \theta_0} & \frac{\partial^2 J^i}{\partial \theta_1^2} \end{pmatrix} = \begin{pmatrix} 2 & 2x^i \\ 2x^i & 2x^{i2} \end{pmatrix} \quad (31)$$

A convex function has the property that its Hessian matrix is positive semidefinite. There are several ways of showing that the Hessian matrix of  $J^i$  is positive semidefinite. A matrix is positive semidefinite if it is symmetric and all its eigenvalues are non-negative, with at least one equal to zero. We proceed to find the eigenvalues of  $H$ :

The equation to solve for finding the eigenvalues is

$$\begin{vmatrix} 2 - \lambda & 2x^i \\ 2x^i & 2x^{i2} - \lambda \end{vmatrix} = 0 \quad (32)$$

This yields

$$\lambda^2 - 2(x^{i2} + 1)\lambda = 0$$

Therefore, the eigenvalues of  $H$  are

$$\lambda = 0, \quad \lambda = 2(x^{i2} + 1) \quad (33)$$

Hence, the Hessian matrix of  $J^i$  is positive semidefinite, and  $J^i$  is a convex function. The sum of two or more convex functions is also a convex function. As the MSE is a sum of all the  $J^i$  (multiplied by a constant) it must be convex.