Linear Regression

August 31, 2020

Abstract

Topics: linear regression model, mean squared error (MSE), gradient descent, stochastic gradient descent, matrix calculus, normal equation.

1 Linear regression model

Linear regression is one of the simple and yet fundamental supervised learning problem in machine learning. A set of m training examples (or instances) $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m$ is given to you, with the superscript (i) indexes the examples. Each training instance has a column feature vector $\mathbf{x}^{(i)} = [x_1^{(i)}, \ldots, x_n^{(i)}]^\top \in \mathbb{R}^n$ is a vector of n real numbers. The superscript $^\top$ means a transposition of a vector or a matrix. $\mathbb{R}^n$ denotes the set (or space) of all length-n vectors of real numbers. The output variable $y^{(i)}$ is a real number associated with the input $\mathbf{x}^{(i)}$. For convenience, we sometimes append a constant 1 to each feature vector so that $\mathbf{x}^{(i)}$ becomes $[1, x_1^{(i)}, \ldots, x_n^{(i)}]^\top \in \mathbb{R}^{n+1}$.

The goal of linear regression is to learn a hypothesis (or function) that is linear in the features of $\mathbf{x}^{(i)}$, so that the function can output a real number $h(\mathbf{x}^{(i)})$ that is as close to $y^{(i)}$ as possible. The linear function $h(\cdot; \boldsymbol{\theta})$ (or $h_{\boldsymbol{\theta}}(\cdot)$) is parametrized by the parameter vector $\boldsymbol{\theta} \in \mathbb{R}^{n+1}$ (or in $\mathbb{R}^n$ is the constant 1 is not appended):

$$h(\mathbf{x}; \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \mathbf{x} = \langle \boldsymbol{\theta}, \mathbf{x} \rangle = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n \tag{1}$$

where $\mathbf{x}$ denotes any feature vector. The notation $\langle \boldsymbol{\theta}, \mathbf{x} \rangle$ means inner product of two vectors.

1.1 Mean squared error

By selecting different parameters $\boldsymbol{\theta}$, $h(\mathbf{x}^{(i)}; \boldsymbol{\theta})$ can be far away or close to the output $y^{(i)}$. To measure the quality of any $\boldsymbol{\theta}$, a quantitative metric is needed. In particular, Mean squared error (MSE) is used as a loss (or cost) function to evaluate a parameter $\boldsymbol{\theta}$:

$$J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \left(h(\mathbf{x}^{(i)}; \boldsymbol{\theta}) - y^{(i)} \right)^2. \tag{2}$$

Note that: 1) $J(\boldsymbol{\theta})$ depends on the training examples as well, but we ignore them to simplify the notation; 2) the fraction $\frac{1}{2}$ is conventional and just scales the square error by a constant; 3) the summation over the m training examples is to pick a $\boldsymbol{\theta}$ that works well on all training examples.

Linear regression selects $\boldsymbol{\theta}$ to minimize the MSE $J(\boldsymbol{\theta})$:

$$\boldsymbol{\theta}^* = \operatorname*{arg\,min}_{\boldsymbol{\theta} \in \mathbb{R}^{n+1}} J(\boldsymbol{\theta}) = \operatorname*{arg\,min}_{\boldsymbol{\theta} \in \mathbb{R}^{n+1}} \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \left(h(\mathbf{x}^{(i)}; \boldsymbol{\theta}) - y^{(i)} \right)^2. \tag{3}$$

The notation "argmin" means that we select an element from the set $\mathbb{R}^{n+1}$ to minimize the function that follows. The function J is a quadratic function in any one coordinate of $\boldsymbol{\theta}$, given the other coordinates fixed as constants. More over, the quadratic function open upwards in a bowl-shape. We call such functions "convex". In fact, the function J is convex in the entire vector $\boldsymbol{\theta}$ [1].

[1] Take the second order derivative of J w.r.t. $\boldsymbol{\theta}$ and verify the resulting Hessian matrix is positive definite.

2 Gradient descent

Since the search space for $\boldsymbol{\theta}$ is the entire set of $(n+1)$-dimensional vectors, we need a efficient way to find the best parameter. At any point $\boldsymbol{\theta}$ in the space $\mathbb{R}^{n+1}$, gradient of the loss function $J(\boldsymbol{\theta})$ gives the direction that one can move to get the steepest descent in the loss function.

The gradient of a scalar function $J(\boldsymbol{\theta})$ with respect to the function's vector argument $(\boldsymbol{\theta})$ is defined as:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \begin{bmatrix} \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_0} \\ \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_n} \end{bmatrix} \tag{4}$$

But $\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j}$ is just a regular derivative of a scalar function with respect to another scale, something you have learned in a calculus course. In particular, for $j = 1, \ldots, n$,

$$\begin{aligned} \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(h(\mathbf{x}^{(i)}; \boldsymbol{\theta}) - y^{(i)} \right)^2 && (5) \\ &= \frac{1}{2m} \sum_{i=1}^{m} \frac{\partial}{\partial \theta_j} \left(h(\mathbf{x}^{(i)}; \boldsymbol{\theta}) - y^{(i)} \right)^2 && (6) \\ &= \frac{1}{2m} \sum_{i=1}^{m} \frac{\partial}{\partial \theta_j} \left(\theta_0 + \theta_1 x_1^{(i)} + \cdots + \theta_n x_n^{(i)} - y^{(i)} \right)^2 && (7) \\ &= \frac{1}{m} \sum_{i=1}^{m} \left(h(\mathbf{x}^{(i)}; \boldsymbol{\theta}) - y^{(i)} \right) x_j^{(i)} && (8) \end{aligned}$$

When $j = 0$,

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} \left(h(\mathbf{x}^{(i)}; \boldsymbol{\theta}) - y^{(i)} \right) \tag{9}$$

2.1 Gradient descent algorithm

Once you have the gradient $\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}|_{\boldsymbol{\theta}}$ evaluated at any point $\boldsymbol{\theta}$, the gradient descent algorithm works as follows:

1. Start from any $\boldsymbol{\theta}^{(0)}$.
2. Iterate through $t = 1, \ldots,$ and update $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} - \alpha \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}|_{\boldsymbol{\theta}^{(t-1)}}$.
3. Stop when $J(\boldsymbol{\theta})$ does not decrease significantly.

In the algorithm, α is a real positive number called "learning rate", which controls how far to go in the direction of $-\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$.

With a proper α, an example run of the gradient descent algorithm when training θ_1, with θ_0 fixed, is shown in Figure 1. In Figure 2, both θ_1 and θ_0 are trained together and the cost function has two parameters. The right subfigure of Figure 2 shows the contour of the cost function in the space of $\mathbb{R}^2 = [\theta_0, \theta_1]$.

The learning rate has to be tuned carefully. If it is too small, the algorithm will not make sufficient progress, while if it is too large, the algorithm will overshot the optimal parameter.

3 Matrix calculus

Differing from iteratively taking gradient descent algorithms mentioned in Section 2.1, one can get the solution for optimal parameters, $\boldsymbol{\theta}^*$, explicitly by **normal equations**. Rather than diving into the normal equations directly, we recall preliminary and define notations first.

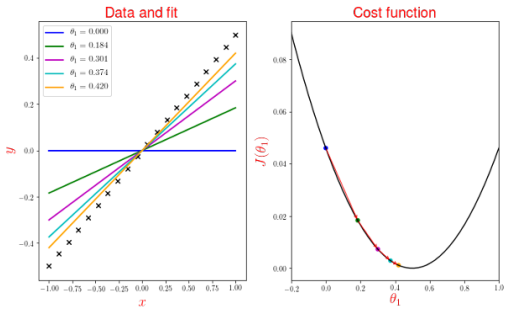

Figure 1: Proper learning rate.

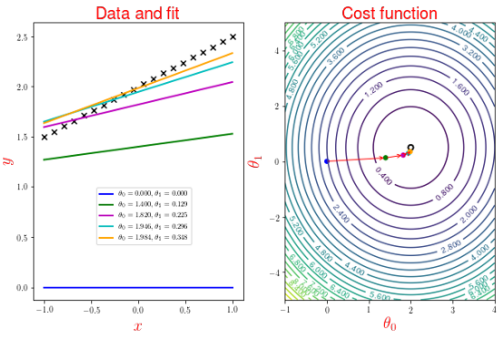

Figure 2: Gradient descent in the contour of the cost function.

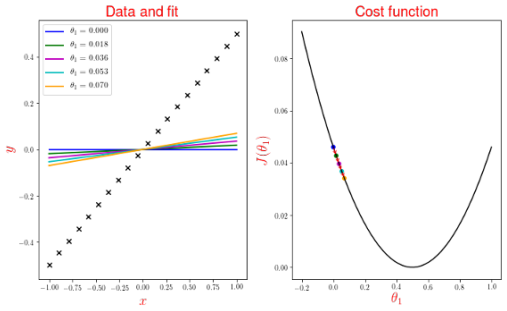

Figure 3: Too small learning rate

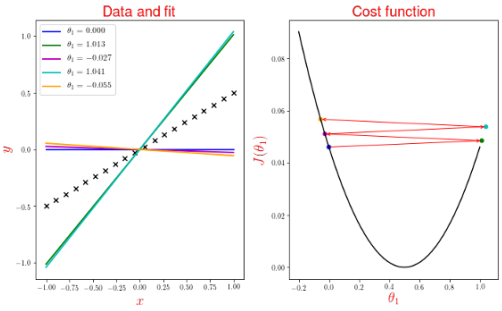

Figure 4: Too large learning rate

3.1 Matrix gradient

Define a function $f(\mathbf{A}) : \mathbb{R}^{n\times m} \mapsto \mathbb{R}$, which maps an $n \times m$ matrix $\mathbf{A}$ to the real domain. Then the gradient of $f(\mathbf{A})$ with respect to $\mathbf{A}$, i.e., $\frac{\partial f(\mathbf{A})}{\partial \mathbf{A}}$, is also an $n \times m$ matrix, whose (i, j)-th element is $\frac{\partial f}{\partial \mathbf{A}_{ij}}$. Specifically,

$$\frac{\partial f(\mathbf{A})}{\partial \mathbf{A}} = \begin{bmatrix} \frac{\partial f}{\partial \mathbf{A}_{11}} & \cdots & \frac{\partial f}{\partial \mathbf{A}_{1m}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial \mathbf{A}_{n1}} & \cdots & \frac{\partial f}{\partial \mathbf{A}_{nm}} \end{bmatrix}. \tag{10}$$

3.2 Traces

The **trace** of the square matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is defined as the sum of elements on the main diagonal, i.e., $\text{tr}(\mathbf{A}) = \sum_{i=1}^{n} \mathbf{A}_{ii}$.

Based on the definition of trace, it is easily to derive its properties:

- For two matrices $\mathbf{A} \in \mathbb{R}^{n\times m}$ and $\mathbf{B} \in \mathbb{R}^{m\times n}$,

$$\text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA}), \tag{11}$$

  notice that $\mathbf{A}$ and $\mathbf{B}$ are *not* necessarily square, e.g., $m \neq n$, but their matrix multiplication, $\mathbf{AB}$ and $\mathbf{BA}$, should be square. It is easy to proof since $\text{tr}(\mathbf{AB}) = \sum_{i=1}^{n}\sum_{j=1}^{m} A_{ij}B_{ji} = \text{tr}(\mathbf{BA})$, Furthermore, as a corollary, $\text{tr}(\mathbf{ABC}) = \text{tr}(\mathbf{CAB}) = \text{tr}(\mathbf{BCA})$, when the matrix multiplication results in square matrix.

- Let $f(\mathbf{A}) = \text{tr}(\mathbf{AB}) : \mathbb{R}^{n\times m} \mapsto \mathbb{R}$, then the gradient of $f(\mathbf{A})$ is:

$$\frac{\partial f(\mathbf{A})}{\partial \mathbf{A}} = \mathbf{B}^\top. \tag{12}$$

  The proof can be derived from the $\text{tr}(\mathbf{AB}) = \sum_{i=1}^{n}\sum_{j=1}^{m} A_{ij}B_{ji}$, and the partial gradient of $f(\mathbf{A})$ with respect to element $\mathbf{A}_{ij}$, $\frac{\partial f}{\partial \mathbf{A}_{ij}}$, is $\mathbf{B}_{ji}$.

- For a real number $a \in \mathbb{R}$,
$$\text{tr}(a) = a. \tag{13}$$
- For any square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$,
$$\text{tr}(\mathbf{A}) = \text{tr}(\mathbf{A}^\top) \tag{14}$$
- $$\nabla_{\mathbf{A}} \text{tr}(\mathbf{A}\mathbf{B}\mathbf{A}^\top\mathbf{C}) = \mathbf{C}\mathbf{A}\mathbf{B} + \mathbf{C}^\top\mathbf{A}\mathbf{B}^\top \tag{15}$$
Just remember or refer to this equation without proving it.

4 Normal equation

Given a training set containing m samples, and each sample has $n+1$ features (including the 1 constant appended to the n features), the **design matrix** $\mathbf{X}$ is defined as an $m \times (n+1)$ matrix and each row is a training sample with an intercept term, e.g., $\mathbf{X} = \begin{bmatrix} —x^{(1)\top}— \\ —x^{(2)\top}— \\ \vdots \\ —x^{(m)\top}— \end{bmatrix}$. The corresponding target values is the column vector $\mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \cdots \\ y^{(m)} \end{bmatrix}$. Now represent the parameters in linear regression model in the column vector $\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \cdots \\ \theta_n \end{bmatrix}$.

By multiplying $\mathbf{X}$ and $\boldsymbol{\theta}$, we have

$$\mathbf{X}\boldsymbol{\theta} = \begin{bmatrix} x^{(1)\top}\boldsymbol{\theta} \\ x^{(2)\top}\boldsymbol{\theta} \\ \vdots \\ x^{(m)\top}\boldsymbol{\theta} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}^\top x^{(1)} \\ \boldsymbol{\theta}^\top x^{(2)} \\ \vdots \\ \boldsymbol{\theta}^\top x^{(m)} \end{bmatrix} = \begin{bmatrix} h_\theta(x^{(1)}) \\ h_\theta(x^{(2)}) \\ \vdots \\ h_\theta(x^{(m)}) \end{bmatrix}, \tag{16}$$

where h_θ is defined in Equation 1.

The difference between the model's predictions $h_\theta(x)$ and target y can be written as:

$$\mathbf{z} = \mathbf{X}\boldsymbol{\theta} - \mathbf{y} = \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ h_\theta(x^{(2)}) - y^{(2)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix} \tag{17}$$

Since $\mathbf{z}^\top\mathbf{z} = \sum_i z_i^2$ for any vector $\mathbf{z}$,

$$\frac{1}{2m}(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^\top(\mathbf{X}\boldsymbol{\theta} - \mathbf{y}) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \tag{18}$$

Compare it with Equation 2, we have the MSE loss function in matrix-vector forms:

$$J(\boldsymbol{\theta}) = \frac{1}{2m}(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^\top(\mathbf{X}\boldsymbol{\theta} - \mathbf{y}). \tag{19}$$

To minimize the loss $J(\boldsymbol{\theta})$, we take the gradient of $J(\boldsymbol{\theta})$ with respect to parameters $\boldsymbol{\theta}$:

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\theta}}(\mathbf{X}\boldsymbol{\theta}-\mathbf{y})^\top(\mathbf{X}\boldsymbol{\theta}-\mathbf{y}) &= \frac{\partial}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^\top\mathbf{X}^\top-\mathbf{y}^\top)(\mathbf{X}\boldsymbol{\theta}-\mathbf{y}) && (20)\\
&= \frac{\partial}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\theta}-\boldsymbol{\theta}^\top\mathbf{X}^\top\mathbf{y}-\mathbf{y}^\top\mathbf{X}\boldsymbol{\theta}+\mathbf{y}^\top\mathbf{y}) && (21)\\
&= \frac{\partial}{\partial \boldsymbol{\theta}}\mathrm{tr}(\boldsymbol{\theta}^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\theta}-\boldsymbol{\theta}^\top\mathbf{X}^\top\mathbf{y}-\mathbf{y}^\top\mathbf{X}\boldsymbol{\theta}) && (22)\\
&= \frac{\partial}{\partial \boldsymbol{\theta}}\mathrm{tr}(\boldsymbol{\theta}\boldsymbol{\theta}^\top\mathbf{X}^\top\mathbf{X}-2\boldsymbol{\theta}^\top\mathbf{X}^\top\mathbf{y}) && (23)\\
&= \mathbf{X}^\top\mathbf{X}\boldsymbol{\theta}+\mathbf{X}^\top\mathbf{X}\boldsymbol{\theta}-2\mathbf{X}^\top\mathbf{y} && (24)\\
&= 2[(\mathbf{X}^\top\mathbf{X})\theta-\mathbf{X}^\top\mathbf{y}]. && (25)
\end{aligned}$$

Third equality holds because because all four terms in the summation are scalars, and $\mathrm{tr}(a)=a$ based on Equation 13. Besides, we ignore $\mathbf{y}^\top\mathbf{y}$ since this term doesn't contain $\boldsymbol{\theta}$.

Forth equality holds because $\mathrm{tr}(\boldsymbol{\theta}^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\theta})=\mathrm{tr}(\boldsymbol{\theta}\boldsymbol{\theta}^\top\mathbf{X}^\top\mathbf{X})$ based on the Equation 11, and the remaining part is because $\mathrm{tr}(\boldsymbol{\theta}^\top\mathbf{X}^\top\mathbf{y})=\mathrm{tr}((\boldsymbol{\theta}^\top\mathbf{X}^\top\mathbf{y})^\top)=\mathrm{tr}(\mathbf{y}^\top\mathbf{X}\boldsymbol{\theta})$ from Equation 14.

Fifth equality holds because of Equation 15, with $\mathbf{A}=\boldsymbol{\theta},\mathbf{B}=\mathbf{I},C=\mathbf{X}^\top\mathbf{X}$, The remaining part comes from Equation 12.

To minimize $J(\boldsymbol{\theta})$, we take the gradient as zero, and we get

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}=2[(\mathbf{X}^\top\mathbf{X})\theta-\mathbf{X}^\top\mathbf{y}]=0, \tag{26}$$

from where we take the closed form of value $\boldsymbol{\theta}$ by

$$\boldsymbol{\theta}=(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}. \tag{27}$$

For a square matrix A that is full rank (or invertible), A^{-1} is the inverse of A so that $AA^{-1}=A^{-1}A=I$, where I is the identity matrix of the same shape as A.

If $\mathbf{X}^\top\mathbf{X}$ is not invertible (or full rank), then pseudo-inverse can be taken in place of the inversion.

5 A probabilistic interpretation of linear regression

We develop a probabilistic interpretation of linear regression for two reasons: first, the interpretation can unify multiple supervised learning models, including linear regression and logistic regression; second, it explains why the MSE loss function is so designed.

Let the noises in predicting the ground truth target $y^{(i)}$ be modeled by some random variables $\epsilon^{(i)}$, which are independently distributed according to a normal (Gaussian) distribution $\epsilon^{(i)}\sim\mathcal{N}(0,\sigma^2)$. All $\epsilon^{(i)}$ has the same zero mean and variance σ^2. Note that the mean and variance does not depend on the data point (the index i). We call such a collection of random variable identically and independently distributed (I.I.D).

Taking into account the noises, the observed targets $y^{(i)}=\boldsymbol{\theta}^\top\mathbf{x}^{(i)}+\epsilon^{(i)}$. In other words, one can predict $y^{(i)}$ using $\boldsymbol{\theta}^\top\mathbf{x}^{(i)}$ but expect some 0-mean Gaussian noise in the predicted value. In other words,

$$\epsilon^{(i)}=y^{(i)}-\boldsymbol{\theta}^\top\mathbf{x}^{(i)}. \tag{28}$$

distributed according to $\mathcal{N}(0,\sigma^2)$. This noise captures the aggregated effect on $y^{(i)}$ of the unknown factors that are not modeled by the features in $\mathbf{x}^{(i)}$.

$y^{(i)}\sim\mathcal{N}(\boldsymbol{\theta}^\top\mathbf{x}^{(i)},\sigma^2)$, since adding a constant $\boldsymbol{\theta}^\top\mathbf{x}^{(i)}$ to a 0-mean Gaussian simply generate a new Gaussian with mean of $\boldsymbol{\theta}^\top\mathbf{x}^{(i)}$, without changing the variance. The probability of observing the target y given the input $\mathbf{x}$ is thus

$$\Pr(y^{(i)}|\mathbf{x}^{(i)};\boldsymbol{\theta})=\frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{(y^{(i)}-\boldsymbol{\theta}^\top\mathbf{x}^{(i)})^2}{2\sigma^2}\right\}. \tag{29}$$

A likelihood function is a function of $\boldsymbol{\theta}$ that parametrized the probability of the observed training data $\{(\mathbf{x}^{(i)},y^{(i)})\}_{i=1}^m$

$$L(\boldsymbol{\theta};\{(\mathbf{x}^{(i)},y^{(i)})\}_{i=1}^m)=\prod_{i=1}^m\Pr(y^{(i)}|\mathbf{x}^{(i)};\boldsymbol{\theta})=\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^m\exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^m(y^{(i)}-\boldsymbol{\theta}^\top\mathbf{x}^{(i)})^2\right\}. \tag{30}$$

we ignore the training data in the likelihood function when the definition is clear in a context and denote the likelihood by $L(\boldsymbol{\theta})$. The product of the probabilities of individual target values is due to the assumption that the m noises $\epsilon^{(i)}$, $i = 1, \ldots, m$, are independent.

5.1 Maximum likelihood estimation

A widely used and important way to train machine learning models is the so-called Maximum likelihood estimation, or MLE for short. MLE chooese $\boldsymbol{\theta}$ to maximize the likelihood $L(\boldsymbol{\theta})$:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}). \tag{31}$$

Since optimization with summations is sometimes easier than with products, we transform the likelihood function to the log-likelihood function:

$$\ell(\boldsymbol{\theta}) = \log L(\boldsymbol{\theta}) = -m \log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2} \sum_{i=1}^{m} (y^{(i)} - \boldsymbol{\theta}^\top \mathbf{x}^{(i)})^2 \tag{32}$$

Since the log function is a monotonically increasing function, taking the log of L does not change the result of the MLE. That is, the $\boldsymbol{\theta}^*$ that maximizes $L(\boldsymbol{\theta})$ also maximizes $\ell(\boldsymbol{\theta})$.

Maximizing $\ell(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ is equivalent to minimize $-\ell(\boldsymbol{\theta}) = -\log L(\boldsymbol{\theta})$. There is a name for the function $-\log L(\boldsymbol{\theta})$: Negative Log-Likelihood (nll) loss function. Minimizing this loss function leads to the same MLE $\boldsymbol{\theta}^*$:

$$\begin{aligned}
\boldsymbol{\theta}^* &= \arg\min_{\boldsymbol{\theta}} -\ell(\boldsymbol{\theta}) && (33)\\
&= \arg\min_{\boldsymbol{\theta}} m \log(\sqrt{2\pi}\sigma) + \frac{1}{2\sigma^2} \sum_{i=1}^{m} (y^{(i)} - \boldsymbol{\theta}^\top \mathbf{x}^{(i)})^2 && (34)\\
&= \arg\min_{\boldsymbol{\theta}} m \log(\sqrt{2\pi}\sigma) + \frac{m}{\sigma^2} J(\boldsymbol{\theta}). && (35)
\end{aligned}$$

We can see the MSE loss function $J(\boldsymbol{\theta})$ for fitting a linear regression model is just a scaled and shifted negative log-likelihood loss. As a result, we have connected the training of linear regression to a the MLE of a probability model.