Supervised Machine Learning

Jiuru Lyu

Contents

1 Linear Regression Model

Basic notations:

- x is the input variable, or **feature**;
- y is the output variable, or **target**;
- m is the number of training examples;
- (x, y) is a single training example;
- $(x^{(i)}, y^{(i)})$ is the i-th training example;
  - Note: the "i" here is an index, rather than exponent.
- $\hat{y}$ is the prediction, or the estimated y;
- f is the model, or hypothesis.

1.1 Representing the Model

Univariate Linear Regression

$$f_{w,b}(x) = wx + b,$$

where w and b are parameters/coefficients/weights

1.2 Cost Function

The Error

$$\hat{y}^{(i)} = f_{w,b}\left(x^{(i)}\right)$$

$$f_{w,b}\left(x^{(i)}\right) = wx^{(i)} + b$$

$$\text{Error} = \hat{y}^{(i)} - y^{(i)}$$

The Squared Error Cost Function (J)

$$\begin{aligned} J(w,b) &= \frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 \\ &= \frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)^2 \end{aligned}$$

The job is the find w and b such that $\hat{y}^{(i)}$ is close to $y^{(i)}$ for all $(x^{(i)}, y^{(i)})$, i.e.,

$$\text{minimize}_{w,b} J(w,b)$$

1.3 Visualizing the Cost Function

In linear regression, the cost function is always a bowl-shaped function. At the bottom of the "bowl", the cost function is minimized.

1.4 Gradient Descent

Outlined Procedure of Gradient Descent

- Start with some w and b
- Keep changing w and b to reduce $J(w,b)$
- Until we settle at or near a minimum.

*Refer to the concept of Gradient ∇f.

1.4.1 Formula

Gradient Descent Algorithm

- Repeat until convergence:

$$w = w - \alpha \frac{\partial}{\partial w} J(w,b)$$

$$b = b - \alpha \frac{\partial}{\partial b} J(w,b)$$

where α is the learning rate.
Key: Simultaneously update w and b.

1.4.2 The Learning Rate

The learning rate cannot be too big or too small.

1.4.3 More on the Formula

Recall:

1. $f_{w,b}(x) = wx + b$
2. $J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}\left(x^{(i)}\right) - y^{(i)}\right)^2$.

Hence,

$$\begin{aligned}
\frac{\partial}{\partial w} J(w,b) &= \frac{\partial}{\partial w}\frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2 \\
&= \frac{\partial}{\partial w}\frac{1}{2m}\sum_{i=1}^{m}\left(wx^{(i)} + b - y^{(i)}\right)^2 \\
&= \frac{1}{2m}\sum_{i=1}^{m}\left(wx^{(i)} + b - y^{(i)}\right)2x^{(i)} \\
&= \frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)x^{(i)}
\end{aligned}$$

$$
\begin{aligned}
\frac{\partial}{\partial b}J(w,b) &= \frac{\partial}{\partial b}\frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2 \\
&= \frac{\partial}{\partial b}\frac{1}{2m}\sum_{i=1}^{m}\left(wx^{(i)} + b - y^{(i)}\right)^2 \\
&= \frac{1}{2m}\sum_{i=1}^{m}\left(wx^{(i)} + b - y^{(i)}\right)2 \\
&= \frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)
\end{aligned}
$$

Therefore, the gradient descent algorithm becomes:

Gradient Descent Algorithm

Repeat until convergence:

$$w = w - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)x^{(i)}$$

$$b = b - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)$$

Update w and b simultaneously.

1.4.4 Gradient Descent in Action

Sometimes, we might encounter cost functions with more than one local minima, indicated by the diagram above. However, in linear regression, we always have a cost function with one and only one local minimum, and that minima is called the global minimum.

2 Linear Model with Multiple Features

2.1 Notation

- x_j is the j-th feature
- n is the number of features
- $\vec{x}^{(i)}$ is a vector representing features of the i-th training example
- $x_j^{(i)}$ is the value of feature j in the i-th training example

2.2 Model

Linear Model with Multiple Features

$$f_{\vec{w},b}(\vec{x}) = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

If we let $\vec{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$ and $\vec{w} = \begin{bmatrix} w_1 & w_2 & \cdots & w_n \end{bmatrix}$ be two column vectors, the model can be written using the **dot product**, as shown below:

Linear Model with Multiple Features using Dot Product

$$f_{\vec{w},b}(\vec{x}) = \vec{w} \cdot \vec{x} + b$$

This is called the multiple linear regression. (We do not call it multivariate regression.)

2.3 Vectorization

By using NumPy, a package in Python, we can easily write parameters and features in an array.

```
w = np.array([1.0,2.5,-3.3])
b=4
x = np.array([10,20,30])
```

There are multiple ways to write the multiple linear regression algorithm in Python, but the most efficient way is to use vectorization.

2.3.1 The least effective way

```
f = w[0] * x[0] +
        w[1] * x[1] +
        w[2] * x[2] + b
```

Note: the code count from '0', so the first index should be '0' instead of '1'.

This way is the least effective way because when we have a large amount of features, we need to write very long and complicated codes to conduct the regression.

2.3.2 A more effective way

```
f = 0
for j in range(0,n):
        f = f + w[j] * x[j]
f = f + b
```

Note: 'range(0,n)' can also be written as 'range(n)', which indicates a range from '0' to 'n', including '0' but excluding 'n'.

In this way, we use the for loop to conduct the regression. This will be faster than the previous way, but less efficient than the vectorization method because when we have a large amount of features, the computer will go through this loop for many times, which is very time-consuming.

2.3.3 The most effective way - Vectorization

```
f = np.dot(w,x) + b
```

In this way, the NumPy directly computes the dot product between the two arrays, w and x. Because it computes the dot product by first multiplying each elements in the array and then adding all the product results together, this method will save time when we are dealing with very large datasets.

2.4 Gradient Descent for Multiple Regression

Cost Function for Multiple Linear Regression

$$J(\vec{w},b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)$$

The Gradient Descent

$$w_j = w_j - \alpha\frac{\partial}{\partial w_j}J(\vec{w},b)$$

$$b = b - \alpha\frac{\partial}{\partial b}J(\vec{w},b)$$

After computing the partial derivative, it becomes

Gradient Descent Algorithm

Repeat until convergence:

$$w_1 = w_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)x_1^{(i)}$$

$$\vdots$$

$$w_n = w_n - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)x_n^{(i)}$$

$$b = b - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)} - y^{(i)}\right)$$

Simultaneously update w_j (for $j = 1,\cdots,n$) and b.

3 Practical Tips for Linear Regression

3.1 Feature Scaling

The size of the feature and parameter will influence the effectiveness of gradient descent. When the range of the feature or the parameter is too large or too small, we should consider to rescale the range so that we can fit them in acceptable ranges. We have several ways to do the feature scaling, including mean normalization and Z-score normalization.

3.1.1 Mean Normalization

Mean Normalizaiton

$$x_n = \frac{x_n - \mu_n}{\max x_n - \min x_n},$$

where μ_n is the mean of x_n.

3.1.2 Z-Score Normalization

Z-Score Normalization

$$x_n = \frac{x_n - \mu_n}{\sigma_n},$$

where μ_n is the mean of x_n, and σ_n is the standard deviation of x_n.

3.2 Checking Gradient Descent for Convergence

We have to ways to make sure the gradient descent is working correctly.

3.2.1 Drawing the diagram of the cost function

The value of our cost function should decrease after ever iteration. We can simply plot the cost function versus number of iterations. If the cost function does decrease as number of iterations increasing, our gradient descent is working correctly.

3.2.2 The Automatics Convergence Test

We can also use the automatic convergence test to tell if our gradient descent is working properly.

Firstly, we need to select an epsilon ε, which is a very small number. For example, we can set $\varepsilon = 10^{-3}$.

If our cost function $J(\vec{w}, b)$ decreases by $\leq \varepsilon$ in one iteration, we declare convergence.

3.3 Choosing the Learning Rate

When we choose the learning rate, we could choose the values from the following array: 0.001, 0.003, 0.01, 0.03, 0.1, 1,... Basically, the next try is three times larger than the previous try.

3.4 Feature Engineering

Feature engineering: Using intuition to design new features by transforming or combining original features.

Example

We have our original model:

$$f_{\vec{w},b}(\vec{x}) = w_1x_1 + w_2x_2 + b,$$

where x_1 represents frontage, and x_2 represents depth.

Now, we can use feature engineering to design a new feature, called area, because area equals to the product between frontage and depth.

$$x_3(\text{area}) = x_1x_2$$

Then, our model is turned into

$$f_{\vec{w},b}(\vec{x}) = w_1x_1 + w_2x_2 + w_3x_3 + b.$$

3.5 Polynomial Regression

We can also have a model with the feature in its different orders.

Examples

$$f_{\vec{w},b}(x) = w_1x + w_2x^2 + w_3x^3 + b;$$
$$f_{\vec{w},b}(x) = w_1x + w_2\sqrt{x} + b$$

4 Classification - Logistic Regression

4.1 Motivations

Binary classification: y can only be one of two values: false (no) or true (yes).

Normally, we record false as 0 and true as 1. 0 is also called the negative class, representing absence; 1 is also called the positive class, indicating presence.

4.2 Logistic Regression

Sigmoid function, or the logistic function, only gives outputs between 0 and 1:

Sigmoid Function

$$g(z) = \frac{1}{1+e^{-z}}, \text{ where } 0 < g(z) < 1.$$

Recall our linear regression function. We can set a function z, which is exactly the linear regression model.

Logistic Model

$$z = \vec{w} \cdot \vec{x} + b$$

Substituting z into our sigmoid function $g(z)$:

$$\begin{aligned} f_{\vec{w},b}(\vec{x}) = g(z) &= g(\vec{w} \cdot \vec{x} + b) \\ &= \frac{1}{1+e^{-(\vec{w} \cdot \vec{x}+b)}}. \end{aligned}$$

This is called the **logistic regression**.

The logistic regression gives the probability that the class is '1'. So, we also denote it as

Logistic Regression

$$f_{\vec{w},b}(\vec{x}) = P(y = 1|\vec{x}; \vec{w}, b)$$

$P(y = 1|\vec{x}; \vec{w}, b)$ represents the probability that y is '1', given input $\vec{x}$ and parameters $\vec{w}$ and b.

Property of Logistic Regreesion

$$P(y = 0) + P(y = 1) = 1.$$

4.3 Decision Boundary

Decision boundary is a boundary that classify data into different classes. We can also regard them as the threshold of transferring from one category to another.

Example

Is $f_{\vec{w},b}(\vec{x}) \geq 0.5$?
Yes: $\hat{y} = 1$; No: $\hat{y} = 0$.

Decision Boundary in Algorithm

When $g(z) \geq 0.5$, $z \geq 0$, $\vec{w} \cdot \vec{x} + b \geq 0$, the regression returns $\hat{y} = 1$.
When $g(z) < 0.5$, $z < 0$, $\vec{w} \cdot \vec{x} + b < 0$, the regression returns $\hat{y} = 0$.

4.4 Cost Function for Logistic Regression

The Cost Function

For logistic regression,

$$f_{\vec{w},b}(\vec{x}) = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{x} + b)}}$$

The cost function is

$$J(\vec{w}, b) = \frac{1}{m} \sum_{i=1}^{m} L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right),$$

where $L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right)$ is the loss function.

Hence, to define a cost function for logistic regression, we need to define the loss function first.

The Loss Function by Definition

$$L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right) = \frac{1}{2}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2$$

Standard Representation of the Loss function

$$L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y^{(i)}\right) = \begin{cases} -\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) & \text{if } y^{(i)} = 1; \\ -\log\left(1 - f_{\vec{w},b}(\vec{x}^{(i)})\right) & \text{if } y^{(i)} = 0. \end{cases}$$

The property of standard representation of the loss function:

- If $y^{(i)} = 1$:
  - As $f_{\vec{w},b}(\vec{x}^{(i)}) \to 1$, then loss $\to 0$;
  - As $f_{\vec{w},b}(\vec{x}^{(i)}) \to 0$, then loss $\to \infty$.
- If $y^{(i)} = 0$:
  - As $f_{\vec{w},b}(\vec{x}^{(i)}) \to 1$, then loss $\to \infty$;
  - As $f_{\vec{w},b}(\vec{x}^{(i)}) \to 0$, then loss $\to 0$.

We can write the loss function in one single function:

Simplied Loss Function

$$L\left(f_{\vec{w},b}(\vec{x}^{(i)}),y^{(i)}\right) = -y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) - (1-y^{(i)})\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right)$$

In this way, when $y^{(i)} = 1$,

$$\begin{aligned}
&L\left(f_{\vec{w},b}(\vec{x}^{(i)}),y^{(i)}\right) \\
&\qquad = -\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) - (1-1)\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right) \\
&\qquad = -\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right)
\end{aligned}$$

When $y^{(i)} = 0$,

$$\begin{aligned}
&L\left(f_{\vec{w},b}(\vec{x}^{(i)}),y^{(i)}\right) \\
&\qquad = -0\times\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) - (1-0)\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right) \\
&\qquad = -\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right)
\end{aligned}$$

Hence, our cost function is written as

Cost Function

$$\begin{aligned}
J(\vec{w},b) &= \frac{1}{m}\sum_{i=1}^{m}\left[L\left(f_{\vec{w},b}(\vec{x}^{(i)}),y^{(i)}\right)\right] \\
&= \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) - (1-y^{(i)})\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right)\right] \\
&= -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) + (1-y^{(i)})\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right)\right]
\end{aligned}$$

4.5 Gradient Descent for Logistic Regression

The gradient descent for logistic regression is similar to that of regression model, expect for different expressions for $f_{\vec{w},b}(\vec{x})$.

Gradient Descent for Logistic Regression

Repeat until convergence:

$$w_j = w_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

$$b = b - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)} - y^{(i)}\right)$$

Simultaneously update w_j (for $j = 1,\cdots,n$) and b.

5 Overfitting and Regularization

5.1 The Problem of Overfitting

- **Underfit**: The model does not fit the training set well, also known as high bias.
- **Generalization**: The model fits the training set pretty well.
- **Overfit**: The model fits the training set extremely well, also known as high variance.

5.2 Addressing Overfitting

Addressing Overfitting

1. Collect more training examples
2. Select features to include/exclude: Feature selection
3. Regularization: Reduce the size of parameters w_j.

5.3 Regularization

To find small values of w_j, we introduce a regularization term to our cost function.

Regularized Cost Function

$$J(\vec{w},b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} b^2,$$

where $\lambda > 0$ is called the regularization parameter.

Normally, we exclude the term relating to the parameter b becuase we do not want be to be extremely small. Hence, the most frequently used regularized cost function is

Regularized Cost Function

$$J(\vec{w},b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2,$$

where

$$\frac{1}{2m}\sum_{i=1}^{m}(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})^2$$

is called the mean squared error, and

$$\frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

is called the regularization term.

To select an appropriate λ, we find the following properties:

- When λ is very big, to minimize $J(\vec{w},b)$, the result will yield very small w_j.
- When λ is relatively small, to minimize $J(\vec{w},b)$, the result will yield larger w_j.

5.4 Regularized Linear regression

Recall: The gradient descent algorithm for linear regression is

The Gradient Descent Algorithm for Linear Regression

Repeat until convergence:

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\vec{w}, b)$$

$$b = b - \alpha \frac{\partial}{\partial b} J(\vec{w}, b)$$

Simultaneously update w_j (for $j = 1, \cdots, n$) and b.

Substitute the regularized cost function, we get:

The Regularized Gradient Descent Algorithm for Linear Regression

Repeat until convergence:

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} \left[\frac{1}{2m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^{n} w_j^2 \right]$$

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} \left[\frac{1}{2m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^{n} w_j^2 \right]$$

Simultaneously update w_j (for $j = 1, \cdots, n$) and b.

Computing the partial derivatives, we get:

$$\begin{aligned}
\frac{\partial}{\partial w_j} J(\vec{w}, b) &= \frac{\partial}{\partial w_j} \left[\frac{1}{2m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^{n} w_j^2 \right] \\
&= \frac{1}{2m} \sum_{i=1}^{m} \left[(\vec{w} \cdot \vec{x}^{(i)} + b - y^{(i)}) 2 x_j^{(i)} \right] + \frac{\lambda}{2m} 2 w_j \\
&= \frac{1}{m} \sum_{i=1}^{m} \left[(\vec{w} \cdot \vec{x}^{(i)} + b - y^{(i)}) x_j^{(i)} \right] + \frac{\lambda}{m} w_j \\
&= \frac{1}{m} \sum_{i=1}^{m} \left[(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}) x_j^{(i)} \right] + \frac{\lambda}{m} w_j \\
\frac{\partial}{\partial b} J(\vec{w}, b) &= \frac{\partial}{\partial b} \left[\frac{1}{2m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^{n} w_j^2 \right] \\
&= \frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)} - y^{(i)} \right)
\end{aligned}$$

Substituting the partial derivatives to the gradient descent algorithm, we get:

The regularized gradient descent algorithm for linear regression

Repeat until convergence:

$$w_j = w_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left[(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}) x_j^{(i)} \right] + \frac{\lambda}{m} w_j \right];$$

$$b = b - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right).$$

Simultaneously update w_j (for $j = 1, \cdots, n$) and b.

5.5 Regularized Logistic Regression

The regularized cost function for logistic regression is given by

The Regularized Cost Function for Logistic Regression

$$J(\vec{w}, b) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log \left(f_{\vec{w},b}(\vec{x}^{(i)}) \right) + (1 - y^{(i)}) \log \left(1 - f_{\vec{w},b}(\vec{x}^{(i)}) \right) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} w_j^2$$

The gradient descent algorithm for regularized logistic regression looks the same, except for $f_{\vec{w},b}(\vec{x}^{(i)})$ representing the logistic regression model.