Linear Regression & Gradient Descent

Jerry Peng

1 Supervised & Unsupervised Learning

Supervised Learning The model is trained by learning the association between input data (features) and its corresponding output data (labels). In this training, the goal of the model is to "learn" the mapping from input to output by minimizing the gap between predicted values and actual values. Then, the model can be used to predict the output of new, unlabeled data. Common supervised learning tasks include classification (for example, determining whether an email is spam or not) and regression (for example, predicting house prices).

Unsupervised Learning The model only has input data (features) to learn from, without associated output labels. The goal of the model is to discover the structure or patterns in the input data. Common unsupervised learning tasks include clustering (for example, dividing customers into several different groups, with customers in each group having similar purchasing habits) and dimensionality reduction (for example, PCA).

How to tell? When determining whether a Machine Learning Algorithm is Supervised Learning, one needs to find out if the dataset used for training is in the form of $feature_i \to label_i$

2 Linear Regression

Regression Regression is a statistical and machine learning method used to understand and predict the relationship between two or more variables. In regression analysis, there is usually one dependent variable (also called the target variable or response variable) and one or more independent variables (also known as features or predictor variables). The goal of regression is to establish a model that describes the relationship between the dependent and independent variables. This model can be used to predict new, unknown data.

2.1 Definations

The simplest and most common type of regression is linear regression, which assumes a linear relationship between the dependent and independent variables. The form of the linear regression model is as follows:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \dots \beta_n x_n + \epsilon \tag{1}$$

β is weight, ϵ is the error term or noise. The formula from above can be rewritten as:

$$y = wx + b \tag{2}$$

or:

$$X\theta = y \tag{3}$$

X is the set of the features, y is the set of the labels, θ is the set of the weights.

The loss/cost function for linear regression is

$$J(\theta) = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y} - y)^2$$

where y is the prediction results from the linear regression model, and $\hat{y}$ is the ground truth values for the labels.

2.2 Solve for θ

We can use Ordinary Least Squares (OLS) to find the value of θ. The goal is to minimize the sum of squared residuals (SSR) between the predicted values of the model and the actual values. The detailed derivation is as follows: Suppose we have a Linear Regression, expressed as:

$$f(x) = X\theta + \epsilon \tag{4}$$

Then we can get SSR base on the definition:

$$\begin{aligned} SSR &= (y - X\theta)^T(y - X\theta) \\ &= (y^T - \theta^T X^T)(y - X\theta) \\ &= y^T y - y^T X\theta - \theta^T X^T y + \theta^T X^T X\theta \end{aligned} \tag{5}$$

Next, we find the derivative of SSR with respect to θ:

$$\begin{aligned} \frac{\partial SSR}{\partial \theta} &= 0 - X^T y - X^T y + 2X^T X\theta \\ &= -2X^T y + 2X^T X\theta \end{aligned} \tag{6}$$

Note: during the differentiation process, we use the following formula:

$$\frac{\partial (X^T X\theta)}{\partial \theta} = X^T X \tag{7}$$

$$\frac{\partial (y^T X\theta)}{\partial \theta} = X^T y \tag{8}$$

Setting the value of the derivative to zero, we get the following derivation:

$$\begin{aligned} \frac{\partial SSR}{\partial \theta} &= 0 \\ -2X^T y + 2X^T X\theta &= 0 \\ X^T y &= X^T X\theta \\ \theta &= (X^T X)^{-1} X^T y \end{aligned} \tag{9}$$

Then we can get the formula of θ:

$$\theta = (X^T X)^{-1} X^T y \tag{10}$$

3 Gradient Descent

Gradient Descent is an optimization algorithm used to solve machine learning and deep learning models, especially in situations where direct solutions are not feasible. Its goal is to minimize the loss function by iteratively updating the model parameters. Mathematically, the gradient represents the directional derivative of a function at a certain point, which is the direction in which the function changes most rapidly. In the gradient descent algorithm, we update the parameters in the direction of the negative gradient because this direction results in the fastest decrease in the function value. The basic steps of the gradient descent algorithm are as follows:

1. Initialize the model parameters (can be randomly initialize).
2. Calculate the gradient of the loss function. The gradient is the partial derivative of the loss function with respect to the model parameters.
3. Update the model parameters in the direction of the negative gradient. The magnitude of the update is determined by the learning rate (a preset positive number). The formula to update the model parameters θ is:

   $$\theta_j := \theta_j - \alpha \times \nabla J(\theta)$$

   where α is the learning rate set up by the user, $\nabla J(\theta)$ is the gradient
4. Repeat steps 2 and 3 until the termination criteria are met, such as when the gradient is close to 0 (i.e., a local minimum is found), or when a preset maximum number of iterations is reached.

3.1 Gradient Descent for Linear Regression

For the linear regression we introduced above, we can also use gradient descent to find the optimal θ. As we mentioned it before, the loss/cost function for the linear regression is

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^{n} (\hat{y} - y)^2$$

Note: Why do we use $\frac{1}{2m}$ and $\frac{\lambda}{2m}$ at here? Let's see the loss expression without $\frac{1}{2m}$:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Taking the derivative with respect to θ_j, we get:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Now let's see the MSE expression with $\frac{1}{2m}$:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Take the derivative:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

At this point, the power of 2 in the power rule will be canceled out by the factor of $\frac{1}{2}$, so the two formulas after differentiation are the same. Therefore, in practice, we often use $\frac{1}{2m}$. The main reason for this is for mathematical convenience. Since the power of 2 is canceled out, including the $\frac{1}{2}$ factor can simplify the derivative. This makes the equation clearer and simplifies the calculation.

From above we calculated the derivative/gradient $\nabla J(\theta)$, and now we can update the θ, which will be updated by using the formula:

$$\theta_j := \theta_j - \alpha \times \nabla J(\theta)$$

At here we finished one step of gradient descent. Now we will repeat this algorithm until the loss converges or the training epochs are reached.

References