Linear Regression

Baboo J. Cui

May 22, 2019

Approximate output y as a function of inputs(features) x, note that both x and y can be vectors.

$$\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots$$

- parameters(weights): θ_i, which control the mapping from $\mathcal{X}$ to $\mathcal{Y}$.
- subscript notation: h_θ means function h is parametrized by θ
- hat notation: $\hat{}$ represent the estimation

By convention, set $x_0 = 1$, so that

$$\hat{y} = h_\theta(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x = x^T \theta$$

- intercept term: x_0 here is known as intercept term
- vector dimension: by default, all vectors are column vectors
- the number of input is n instead of $n+1$, the number of parameter is $n+1$

Cost function is defined as

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

- $J(\theta)$ is known as least-square cost function, lead to ordinary least square regression model
- m is the length of training list

We need to optimize θ to minimize $J(\theta)$.

1 Least Mean Square(LMS) Algorithm

Gradient descent algorithm can find a value in domain to minimize a function, for general scalar function $y = f(x)$, by setting

$$x := x - \alpha \frac{d}{dx} f(x)$$

a value of x can be found to minimize $f(x)$, note that:

- $:=$ represent assignment
- x can be a vector, if so, take partial derivative to all variables(next part)
- α is called rate(or learning rate in ML), it is positive
- gradient is the deepest increase direction, so its negative is the deepest decrease direction, in this case, the gradient is a scalar

For vector case, let's say $y = f(\mathbf{x})$, where $x \in \mathbb{R}^n$, the gradient of function f is denoted by ∇f, and is defined as

$$\nabla f = \nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}^T$$

the minimization process can be compactly written as

$$\mathbf{x} := \mathbf{x} - \alpha \nabla f(\mathbf{x})$$

To apply gradient descent method to $J(\theta)$

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

to get ∇J, use chain rule for each element

$$\begin{aligned} \frac{\partial J(\theta)}{\partial \theta_j} &= \frac{\partial \quad \frac{1}{2}\sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2}{\partial \theta_j} \\ &= 2 \times \frac{1}{2} \sum_{i=1}^m \left((h_\theta(x^{(i)}) - y^{(i)}) \frac{\partial (h_\theta(x^{(i)}) - y^{(i)})}{\partial \theta_j} \right) \\ &= \sum_{i=1}^m \left((h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \right) \end{aligned}$$

- this is the LMS update rule
- this is called batch gradient descent since it looks every example, bad computational cost
- this algorithm is sensitive to initial condition, may lead to different local minimum. it always converge to global minimum if function is convex
- J is a convex quadratic function, which is good

Instead of looking for all element, if the parameter is updated just based on new coming example, the cost function can be simplified as

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

and each update will be simpler. This is known as stochastic gradient descent method.

- stochastic algorithm is computationally faster than batch gradient
- stochastic algorithm coverages faster than batch gradient
- stochastic algorithm may oscillate and never reach the minimum
- stochastic algorithm is preferred when training set is large

2 Normal Equation

Normal equation can explicitly find the solution to LMS problem by setting derivative of cost function to zero. Define design matrix

$$X \in \mathbb{R}^{m\times(n+1)}$$

- each row of X represents n features of one training example with intercept 1, all together $n+1$ element
- the number of row of X represent m training examples

Let y be the target value set so that

$$y \in \mathbb{R}^{m\times 1}$$

Let θ be the parameter set so that

$$\theta \in \mathbb{R}^{(n+1)\times 1}$$

It should be clear that

$$J(\theta) = \frac{1}{2}(X\theta - y)^T(X\theta - y)$$

By properties of matrix we have

$$\nabla_\theta J(\theta) = X^TX\theta - X^Ty$$

Set it to be zero and we can get normal equation

$$X^TX\theta = X^Ty$$

So the closed form of θ can be found as

$$\theta = (X^TX)^{-1}X^Ty$$

This is also known as pseudo inverse in linear algebra.

3 Probabilistic Interpretation

Suppose target variable and inputs are related via

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$$

where $\epsilon^{(i)}$ are the errors and are independently and identically distributed(IID) as

$$\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$$

Recall multivariate normal distribution is

$$x \sim \mathcal{N}(\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^k|\Sigma|}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

where Σ is the covariance matrix, which is symmetric and positive definite. So the density function of ϵ is given by

$$p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}\right)$$

This implies that

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

which is equivalent to

$$\left(y^{(i)}|x^{(i)};\theta\right) \sim \mathcal{N}(\theta^T x^{(i)}, \sigma^2)$$

Bringing back design matrix X, and the parameter θ, the probability of y is denoted as

$$p(y|X;\theta)$$

Note that in this case, a bad θ will cause horrible deviation from y to $\hat{y}$. When we wish to explicitly view this as a function of θ, it is called likelihood function:

$$L(\theta) = L(\theta; X, y) = p(y|X;\theta)$$

Based on the assumption on $\epsilon^{(i)}$, L can be written as:

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta) \\ &= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \end{aligned}$$

Clearly, from the principle of maximum likelihood, θ should be chosen to maximize $L(\theta)$. Maximizing a function can be achieved by maximizing any strictly increasing function of that function, and we choose to take log likelihood $l(\theta)$:

$$\begin{aligned} l(\theta) &= \ln L(\theta) \\ &= \ln \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\ &= \sum_{i=1}^{m} \ln \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\ &= m \ln \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2}\frac{1}{2}\sum_{i=1}^{m}(y^{(i)} - \theta^T x^{(i)})^2 \end{aligned}$$

to maximize this equation is to minimizing

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

which is the cost function $J(\theta)$, the origin least square cost function. And the we can summarize that

- least square is equivalent to maximizing likelihood estimation(why it is natural)
- the probabilistic assumption is not necessary
- the variance σ^2 doesn't matter in this case

4 Locally Weighted Linear Regression

for next week...