Machine Learning

Nagubandi Krishna Sai
MS20BTECH11014

June 2021

Contents

Chapter 1

Fundamentals of Machine Learning

1.1 Supervised Learning

Supervised Learning gives ”correct answers”, the output values are same as real life values.
In Supervised Learning, we are given a set of data and we know what our correct output should look like, having an idea that there is relationship between the input and the output.
Supervised Learning problems has two types of problems,

1. Regression.
2. Classification.

1.1.1 Linear Regression

In regression type of problems, we are trying to predict results within a continuous output, means that we are trying to map input variables to some continuous function.
Linear regression has real-valued output, but the output will be same or near valued to the actual output.

1. m = Number of training examples.
2. x's = ”input” variable (or) feature.
3. y's = ”output (or) target” variable.
4. (x,y) = one training example.
5. $(x^{(i)},y^{(i)}) = i^{th}$ training example.

Example :

Size of feet^2(x)	Price($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
⋮	⋮

Table 1.1: Training set of housing prices.

1. m = 47.
2. $x^{(1)} = 2104$ and $y^{(1)} = 460$.
3. $x^{(2)} = 1416$ and $y^{(2)} = 232$.

$$\begin{tikzcd}
\boxed{\text{Training set}} \arrow[d] & \boxed{\text{Training set is feeded to learning algorithm}} \\
\boxed{\text{Learning algorithm}} \arrow[d] & \\
\boxed{\text{h}} & \boxed{\text{(hypothesis, This is a function that takes input x and estimate value of y.)}}
\end{tikzcd}$$

1.1.2 Hypothesis

$$h_\theta(x) = \theta_0 + \theta_1 . x \tag{1.1}$$

θ_i's = parameters.

This type of hypothesis mode is called "Linear regression with one variable" (or) "Univariate linear regression."

1.1.3 Cost function

Cost function helps us know that how well to fit the best possible straight line over the given data.

Q$\rangle$ *How to choose* θ_i's ?

1. Choose θ_i 's so that $h_\theta(x)$ is close to y for our training example (x,y).

2. minimise θ_0,θ_1 so, that [$h_\theta(x)$ - y] is small.

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}[h_\theta(x^{(i)}) - y^{(i)}]^2 \tag{1.2}$$

J(θ_0,θ_1) = Cost function (or) Squared error function.

1.1.4 Gradient descent

Gradient descent is used to minimise cost function(J) in linear regression.
Gradient descent is used in many areas to minimise many functions in ML/AI.
Gradient descent algorithm,

$$Repeat\ until\ convergence\ (minimum)\left\{\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta_0, \theta_1),\ for\ j = 0, j = 1.\right. \tag{1.3}$$

1. := is Assignment operator.

2. α is learning rate.

3. $\frac{\partial}{\partial\theta_j}J(\theta_0, \theta_1)$ is derivative.

Gradient descent is nothing but the derivative of the Cost function.

$$Slope\ of\ cost\ function\ curve = \frac{\partial J(\theta_1)}{\partial\theta_1},\ when\ \theta_0 = 0. \tag{1.4}$$

Learning rate,

1. If α is too small, gradient descent can be slow. After many such operations(can be infinite times), the 'θ_1' could reach "global minimum".

2. If α is too large, gradient descent can overshoot the minimum. It may "fail to converge (or) even diverge".

3. If θ_1 is at the local optima itself when we started or taken θ_1, then there is no use of "α (or) gradient descent".

1.1.5 Linear Regression for multivariables

Hypothesis,

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n. \tag{1.5}$$

In the total context of Supervised learning, hypothesis is just predicting the output.
For convenience of notation, declare x$_0$ = 1 ($x_0^{(i)} = 1$).

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \epsilon\ \Re^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \epsilon\ \Re^{n+1}$$

The above matrix is 0 - indexed.

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n . h_\theta(x) = \theta^\top x. \tag{1.6}$$

Cost function,

$$J(\theta) = J(\theta_0, \theta_1, \theta_2, ..., \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} [h_\theta(x^{(i)}) - y^{(i)}]^2 \tag{1.7}$$

Gradient descent,

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, \theta_2, ..., \theta_n) \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} [h_\theta(x^{(i)}) - y^{(i)}] x_j^{(i)} \tag{1.8}$$

Feature scaling,
Get every feature into approximately -1$\leq x_i \leq 1\ range.$
Mean normalization,
$Replace x_i\ with\ x_i - \mu_i\ to\ make\ features\ have\ approximately\ zero\ mean.$
polynomial regression,