Regression Analysis

S. R. M. Prasanna

Dept of Electrical Engineering
Indian Institute of Technology Dharwad

prasanna@iitdh.ac.in

November 23, 2020

Acknowledgements

- Artificial Neural Networks by Prof. B. Yegnanarayana, PHI, 1999
- Machine learning video lectures by Prof. Andrew Ng, Stanford Uty
- Jason Brownlee "Basics of Linear Algebra for Machine Learning", Online book, 2018.

Terminologies in Regression

- Regression: Finding mapping function between given input variable(s) and **output variable which is continuous**.
- Mathematically, given $(\mathrm{x}, \mathrm{y}) = \{(x^{(i)}, y^{(i)})_{i=1,2,\ldots,M}\}$, find mapping function $h_w()$, such that $y = h_w(x)$.
- Using $h_w()$ for unseen data, $\hat{y}_k = h_w(x_k), k \neq i$ and $\hat{y}_k \approx y_k$.
- Univariate vs Mulitvariate: x is 1D vs 2D or more.
- Univariate implies regression with one variable. $y = w_0 + w_1 x$
- Multivariate $\implies y = w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)}$
- Linear vs non-linear: coefficients are linear vs nonlinear.
- Univariate Linear Regression: $y = w_0 + w_1 x$
- Univariate Non-Linear Regression: $y = w_0 + (w_1)^2 x$ or $y = exp(w_0 + w_1 x)$

Univariate Linear Regression

- $y = h_w(x) = w_0 + w_1 x$
- Need to estimate w_0, w_1 which are best fit for the given data (x, y).
- Estimate of y: $\hat{y}_i = h_w(x_i)$.
- Error: $e_i = \hat{y}_i - y_i$.
- Squared Error: $e_i^2 = (\hat{y}_i - y_i)^2$.
- Mean Squared Error (MSE): $E = \frac{1}{M} \sum_{i=1}^{M} e_i^2 = (\hat{y}_i - y_i)^2$
- Finally, (w_0, w_1) that result in least MSE are chosen as parameters.

Cost Function (C)

- $\min_{w_0, w_1} C(w_0, w_1) = \min_{w_0, w_1} \frac{1}{2M} \sum_{i=1}^{M} (w_0 + w_1 x_i - y_i)^2.$

- How to minimize $C(w_0, w_1)$?

- Wrt to single variable, say w_1, cost function i.e., error plot is a **parabola**

- Wrt to two variables, cost function is a **contour** in 2D

- Wrt to three variables, cost function is a **surface** in 3D

- All of them will be convex in nature

- Min point in convex surface will give optimal values for (w_0, w_1)

Gradient Descent Method for Minimization

- More generic method for minimization
- Given function $G(\theta_0, \theta_1)$
- Start with some (θ_0, θ_1)
- Iteratively change (θ_0, θ_1) to minimize $G(\theta_0, \theta_1)$ until minima is reached.
- Values of (θ_0, θ_1) at minima are chosen as optimal parameters.

Gradient Descent Algorithm

- $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} G(\theta_0, \theta_1)$, $j = 0, 1$
- α is a learning rate parameter.
- Partial derivative essentially represents slope
- $\theta_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} G(\theta_0, \theta_1)$
- $\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} G(\theta_0, \theta_1)$
- Repeat the above two steps until convergence
- Smaller α, smaller step size
- Bigger α, bigger step size
- Bigger step size may lead to non-convergence to minima.
- Also large α in some case may diverge.

Gradient Descent for Linear Regression

- Univariate LR: $\min_{w_0, w_1} C(w_0, w_1) = \min_{w_0, w_1} \frac{1}{2M} \sum_{i=1}^{M} (w_0 + w_1 x_i - y_i)^2.$
- Gradient Descent: $G(\theta_0, \theta_1)$
- $\theta_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} G(\theta_0, \theta_1)$
- $\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} G(\theta_0, \theta_1)$
- Repeat the above two steps until convergence
- $\theta_0 = w_0$, $\theta_1 = w_1$, and $G(\theta_0, \theta_1) = C(w_0, w_1)$

Gradient Descent for Linear Regression

- $\frac{\partial}{\partial w_0} C(w_0, w_1) = \frac{\partial}{\partial w_0} \frac{1}{2M} \sum_{i=1}^{M} (w_0 + w_1 x_i - y_i)^2$
- $\frac{\partial}{\partial w_0} C(w_0, w_1) = \frac{1}{M} \sum_{i=1}^{M} (w_0 + w_1 x_i - y_i)$
- $\frac{\partial}{\partial w_1} C(w_0, w_1) = \frac{\partial}{\partial w_1} \frac{1}{2M} \sum_{i=1}^{M} (w_0 + w_1 x_i - y_i)^2$
- $\frac{\partial}{\partial w_1} C(w_0, w_1) = \frac{1}{M} \sum_{i=1}^{M} (w_0 + w_1 x_i - y_i) x_i$
- w_0, w_1 by Gradient Descent:
- $w_0 := w_0 - \alpha \frac{1}{M} \sum_{i=1}^{M} (w_0 + w_1 x_i - y_i)$
- $w_1 := w_1 - \alpha \frac{1}{M} \sum_{i=1}^{M} (w_0 + w_1 x_i - y_i) x_i$
- Repeat the above two steps until convergence

Salary Prediction by Univariate LR

- Let x is experience in months and y is salary in rupees.
- Given $M = 100$ values of (X, Y).
- Learn mapping between salary (y) and experience in month (x): $y = h_w(x)$.
- Prediction of salary for a new x_k using using: $\hat{y}(k) = h_w(x_k)$.
- Expect $\hat{y}(k) \approx y_k$
- Can I better fit the model? Yes, using higher order polynomial in x.
- $y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$

Motivation for Multivariate Linear Regression

- Consider salary prediction problem discussed earlier.
- Let represent x represent total experience.
- Of course, can better fit the curve using higher order polynomial of x
- However, experience has many components like teaching (x_1), research (x_2), admin (x_3) and so on.
- Thus, in univariate LR $x = \{x_1 + x_2 + x_3\}$.
- Each component represent different aspect. Can I do better prediction using different components explicitly ?
- Hence the motivation for multivariate linear regression.

Multivariate Linear Regression

- Univariate LR: $y = h_w(x) = w_0 + w_1 x$
- Multivariate LR: $y = h_w(x) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_N x_N$
- Multivariate LR: $y = h_w(x) = w_0 x_0 + w_1 x_1 + w_2 x_2 + \ldots + w_N x_N$, where $x_0 = 1$
- $h_w(x) = w^T x$, inner product of transpose of parameter vector with input vector
- $w = \{w_0, w_1, \ldots, w_N\}^T$ and $x = \{x_0, x_1, \ldots, x_N\}^T$, w, x, each is col vector of $(N+1)$ dimension

Multivariate Linear Regression using Gradient Descent

- (X, Y), where, $X = \{x_j^{(i)} i = 1, 2, \ldots, M, j = 0, 1, 2, \ldots, N\}$, X is $M \times (N+1)$ matrix and $Y = \{y^{(1)}, y^{(2)}, \ldots, y^{(M)}\}$ is M dim col vector.
- Estimate $w = \{w_0, w_1, w_2, \ldots, w_N\}$ which are best fit for (X, Y).
- $\min\limits_{w_0, w_1, \ldots, w_n} C(w_0, w_1, \ldots, w_N) = \min\limits_{w_0, w_1, \ldots, w_n} \frac{1}{2M} \sum_{i=1}^{M} (y^{\hat{(i)}} - y^{(i)})^2.$
- $\min\limits_{w} C(w) = \min\limits_{w} \frac{1}{2M} \sum_{i=1}^{M} (w^T x^{(i)} - y^{(i)})^2.$

Gradient Descent for Linear Regression

- Gradient Descent: $G(\theta_0, \theta_1, \ldots, \theta_n) = G(\theta)$
- $\theta_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} G(\theta)$
- $\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} G(\theta)$
- $\theta_N := \theta_N - \alpha \frac{\partial}{\partial \theta_N} G(\theta)$
- Repeat the above N steps until convergence
- $\theta_0 = w_0$, $\theta_1 = w_1$, and $\theta_N = w_N$ $G(\theta) = C(w)$

Gradient Descent for Linear Regression

- $\frac{\partial}{\partial w_0} C(w) = \frac{\partial}{\partial w_0} \frac{1}{2M} \sum_{i=1}^{M} (w^T x^{(i)} - y^{(i)})^2$
- $\frac{\partial}{\partial w_0} C(w) = \frac{1}{M} \sum_{i=1}^{M} (w^T x^{(i)} - y^{(i)})$
- $\frac{\partial}{\partial w_1} C(w) = \frac{\partial}{\partial w_1} \frac{1}{2M} \sum_{i=1}^{M} (w^T x^{(i)} - y^{(i)})^2$
- $\frac{\partial}{\partial w_1} C(w) = \frac{1}{M} \sum_{i=1}^{M} (w^T x^{(i)} - y^{(i)}) x_1^{(i)}$
- $\frac{\partial}{\partial w_N} C(w) = \frac{1}{M} \sum_{i=1}^{M} (w^T x^{(i)} - y^{(i)}) x_N^{(i)}$
- w by Gradient Descent:
- $w_0 := w_0 - \alpha \frac{1}{M} \sum_{i=1}^{M} (w_0 + w_1 x^{(i)} - y^{(i)}) x_0^{(i)}$
- $w_1 := w_1 - \alpha \frac{1}{M} \sum_{i=1}^{M} (w_0 + w_1 x^{(i)} - y^(i)) x_1^{(i)}$
- $w_N := w_N - \alpha \frac{1}{M} \sum_{i=1}^{M} (w^T x^{(i)} - y^{(i)}) x_N^{(i)}$
- Repeat the above $(N + 1)$ steps until convergence

Salary Prediction by Multivariate LR

- Let $x = \{x_1, x_2, x_3\}$ represent different experience in months and y represent salary in rupees.
- Given $M = 100$ values of (X, Y), X is a matrix.
- Learn mapping between salary (y) and experience in month (x): $y = h_w(\mathrm{x})$.
- Prediction of salary for a new x_k using using: $\hat{y}(k) = h_w(\mathrm{x}_k)$.
- Expect $\hat{y}(k) \approx y_k$
- Can I better fit the model? Yes, using higher order polynomial in x.
- $y = h_w(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + \ldots$

Parameter Estimation using Normal Equations Process

- $\mathrm{y} = X\mathrm{w}$
- $e = X\mathrm{w} - \mathrm{y}$
- $E = (X\mathrm{w} - \mathrm{y})^T(X\mathrm{w} - \mathrm{y})$
- $E = ((X\mathrm{w})^T - \mathrm{y}^T)(X\mathrm{w} - \mathrm{y})$
- $E = (X\mathrm{w})^T X\mathrm{w} - \mathrm{y}^T X\mathrm{w} - (X\mathrm{w})^T\mathrm{y} + \mathrm{y}^T\mathrm{y}$
- $E = (X\mathrm{w})^T X\mathrm{w} - 2(X\mathrm{w})^T\mathrm{y} + \mathrm{y}^T\mathrm{y}$
- $E = \mathrm{w}^T X^T X\mathrm{w} - 2\mathrm{w}^T X^T\mathrm{y} + \mathrm{y}^T\mathrm{y}$
- $\frac{\partial}{\partial w}E = 0$
- $\frac{\partial}{\partial w}E = 2X^T X\mathrm{w} - 2X^T\mathrm{y} = 0$
- $X^T X\mathrm{w} = X^T\mathrm{y}$
- $\mathrm{w} = (X^T X)^{-1}X^T\mathrm{y}$

Linear Regression using Normal Equations

- Normal equations to find w, than iterative in Gradient descent.
- Method to solve equations rather than iterative procedure.
- Solving Normal Equations: Partially derivative of E or $C(w)$ with each dimension of w equating the resulting derivative eqn to zero.
- Results in $(N+1)$ normal equations.
- Feature matrix: X, where each row is $N+1$ feature vector. There are M such feature vectors.
- y is M dimension output vector and $\mathrm{y} = X\mathrm{w} \implies X^T\mathrm{y} = X^TX\mathrm{w}$
- $\mathrm{w} = (X^TX)^{-1}X^T\mathrm{y}$.

LR: Gradient Descent(GD) vs Normal Equations (NE)

- GD requires choosing α where as NE does not need.
- GD is iterative vs NE is one step.
- GD works well even when N, feature dim is very large.
- NE becomes difficult due to difficulty in finding $(X^TX)^{-1}$.
- If feature dim is small to moderate dim then use NE.
- If feature dim is very large dim then use GD.