

Regression Analysis

Arvinder Singh

March 2023

Contents

1	Linear Regression	2
1.1	Problem Setup	2
1.2	Model	2
1.3	Loss Function	3
1.4	Cost Function	3
1.5	Training	3
1.5.1	Direct Solution	3
1.5.2	Gradient Descent	5
1.5.3	Probabilistic Solution	7
1.6	Evaluation	7
2	Logistic Regression	8

Chapter 1

Linear Regression

1.1 Problem Setup

Assume a dataset $\{\mathbf{X}, \mathbf{Y}\}$ where

$$\mathbf{X} = \begin{bmatrix} x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

the discrete or continuous independent variable $\mathbf{X} \in \mathbb{R}^{n \times d}$ and the continuous dependent variable $\mathbf{Y} \in \mathbb{R}^{n \times 1}$, n is the number of training examples and d is the number of features.

The goal is to use a supervised learning approach for a regression application by learning a model hypothesis $h_\beta(x)$ which predicts $\hat{y}^{(i)}$ given $\mathbf{x}^{(i)}$, assuming the relationship between the two is linear.

1.2 Model

For simple (a.k.a. univariate) linear regression, we can model the distribution of the i^{th} example using the following equation:

$$h_\beta(x^{(i)}) = \beta_0 + \beta_1 x_1^{(i)} + \dots + \beta_d x_d^{(i)} = \beta_0 + \sum_{j=1}^d \beta_j x_j^{(i)}$$

where $i = (1, \dots, n)$, $j = (1, \dots, d)$, β_0 (a.k.a. bias or intercept) represents the average of \mathbf{Y} when all $\mathbf{X} = 0$, β_j (a.k.a. coefficient or slope) represents the average effect of one-unit increase of a certain feature j on \mathbf{Y}

The term β_0 makes the function affine.

1.3 Loss Function

In order to quantify how well our model performs, we must define a function that calculates how "far" our model predictions \hat{y} were from the actual label y . We would call this our "loss" function. One way to define our loss is to calculate the distance between the two values. We will call this distance the residual $r^{(i)}$, where

$$r^{(i)} = y^{(i)} - (\beta_0 + \beta_1 x^{(i)})$$

We would like the residuals to be close to zero.

1.4 Cost Function

We would like to find the loss for each example to give us an average "fit" of the model. We can do so by summing the absolute values of our residuals and dividing the sum by the number of examples. The caveat is, absolute values are a non-differentiable functions, which is not desirable. We can instead choose to average the sum of squared residuals instead, keeping the function differentiable, with an added benefit of penalizing larger losses over smaller ones.

Note: Dividing the sum of squared residuals by n gives us a biased estimate of the variance of the unobserved errors. To remove the bias, divide the sum of squared residuals by $df = n - p - 1$ where df is the degrees of freedom, p is the number of parameters being estimated (excluding the intercept)

To find the optimal values for β , we must minimize the sum of the squared residuals. That sum can be divided by n and give us the mean of the squared residuals (a.k.a. Mean Squared Error).

$$\mathcal{E} = \frac{1}{n} \sum_{i=1}^n r_i^2$$

1.5 Training

Now that we have a model, along with a function to evaluate the fit, we would like to find the optimum values of our parameters β that would give us the least average loss. Fortunately, for linear regression problems, there is a direct, closed-form solution that we can employ. We will also take a look at optimizing our parameters using gradient descent.

1.5.1 Direct Solution

In the case of a simple (univariate) linear regression, we know that the minimum cost occurs when the partial derivative of the cost function with respect to the parameters β_0 and β_1 are at 0. We can use this information to estimate our parameters $\hat{\beta}_0$ and $\hat{\beta}_1$.

Note: Recall from single variable calculus that (assuming a function is differentiable) the minimum x^* of a function f has the property that the derivative $\frac{df}{dx}$ is zero at $x = x^*$. Note that the converse is not true: if $\frac{df}{dx} = 0$, then x^* might be a maximum or an inflection point, rather than a minimum. But the minimum can only occur at points that have derivative zero.

Partial derivative of $\hat{\beta}_0$:

$$\begin{aligned}
\frac{\delta \mathcal{E}}{\delta \hat{\beta}_0} &= \frac{\delta}{\delta \hat{\beta}_0} \frac{1}{n} \sum_{i=1}^n \left(y^{(i)} - \hat{y}^{(i)} \right)^2 \\
&= \frac{\delta}{\delta \hat{\beta}_0} \frac{1}{n} \sum_{i=1}^n \left[y^{(i)} - \left(\hat{\beta}_0 + \hat{\beta}_1 x^{(i)} \right) \right]^2 \\
&= \frac{-2}{n} \sum_{i=1}^n \left(y^{(i)} - \hat{\beta}_0 - \hat{\beta}_1 x^{(i)} \right) \\
&= \frac{-2}{n} \left(\sum_{i=1}^n y^{(i)} - n\hat{\beta}_0 - \hat{\beta}_1 \sum_{i=1}^n x^{(i)} \right) \\
&= -2 \left(\frac{n}{n} \hat{\beta}_0 + \hat{\beta}_1 \frac{1}{n} \sum_{i=1}^n x^{(i)} - \frac{1}{n} \sum_{i=1}^n y^{(i)} \right) \\
&= -2 \left(\hat{\beta}_0 + \hat{\beta}_1 \bar{x} - \bar{y} \right)
\end{aligned}$$

Setting partial derivative to 0 and solving for $\hat{\beta}_0$:

$$\begin{aligned}
-2 \left[\hat{\beta}_0 + \hat{\beta}_1 \bar{x} - \bar{y} \right] &= 0 \\
\hat{\beta}_0 &= \bar{y} - \hat{\beta}_1 \bar{x}
\end{aligned}$$

Partial derivative of $\hat{\beta}_1$:

$$\begin{aligned}
\frac{\delta \mathcal{E}}{\delta \hat{\beta}_1} &= \frac{\delta}{\delta \hat{\beta}_1} \frac{1}{n} \sum_{i=1}^n \left(y^{(i)} - \hat{y}^{(i)} \right)^2 \\
&= \frac{\delta}{\delta \hat{\beta}_1} \frac{1}{n} \sum_{i=1}^n \left(y^{(i)} - \left(\hat{\beta}_0 + \hat{\beta}_1 x^{(i)} \right) \right)^2 \\
&= \frac{-2}{n} \sum_{i=1}^n x^{(i)} \left(y^{(i)} - \hat{\beta}_0 - \hat{\beta}_1 x^{(i)} \right) \\
&= \frac{-2}{n} \sum_{i=1}^n \left(x^{(i)} y^{(i)} - (\bar{y} - \hat{\beta}_1 \bar{x}) x^{(i)} - \hat{\beta}_1 x^{(i)2} \right) \\
&= \frac{-2}{n} \left(\sum_{i=1}^n x^{(i)} y^{(i)} - \bar{y} \sum_{i=1}^n x^{(i)} + \hat{\beta}_1 \bar{x} \sum_{i=1}^n x^{(i)} - \hat{\beta}_1 \sum_{i=1}^n x^{(i)2} \right)
\end{aligned}$$

Setting partial derivative to 0 and solving for $\hat{\beta}_1$:

$$\begin{aligned}
&\left(- \sum_{i=1}^n x^{(i)} y^{(i)} + \bar{y} \sum_{i=1}^n x^{(i)} - \hat{\beta}_1 \bar{x} \sum_{i=1}^n x^{(i)} + \hat{\beta}_1 \sum_{i=1}^n x^{(i)2} \right) = 0 \\
\hat{\beta}_1 &\left(\bar{x} \sum_{i=1}^n x^{(i)} - \sum_{i=1}^n x^{(i)2} \right) = \bar{y} \sum_{i=1}^n x^{(i)} - \sum_{i=1}^n x^{(i)} y^{(i)} \\
\hat{\beta}_1 &= \frac{\bar{y} \sum_{i=1}^n x^{(i)} - \sum_{i=1}^n x^{(i)} y^{(i)}}{\bar{x} \sum_{i=1}^n x^{(i)} - \sum_{i=1}^n x^{(i)2}} \\
&= \frac{n \bar{y} \bar{x} - \sum_{i=1}^n x^{(i)} y^{(i)}}{n \bar{x}^2 - \sum_{i=1}^n x^{(i)2}} = \frac{\sum_{i=1}^n x^{(i)} y^{(i)} - n \bar{y} \bar{x}}{\sum_{i=1}^n x^{(i)2} - n \bar{x}^2}
\end{aligned}$$

To simplify:

$$\begin{aligned}
\sum_{i=1}^n x^{(i)} y^{(i)} - n \bar{y} \bar{x} &= \sum_{i=1}^n (x^{(i)} - \bar{x})(y^{(i)} - \bar{y}) := S_{XY} \\
\sum_{i=1}^n x^{(i)2} - n \bar{x}^2 &= \sum_{i=1}^n (x^{(i)} - \bar{x})^2 := S_{XX} \\
\hat{\beta}_1 &= \frac{S_{XY}}{S_{XX}}
\end{aligned}$$

1.5.2 Gradient Descent

The other way to find the optimized values of our parameters is to use an iterative pproach via gradient descent. The basic algorithm of gradient descent is as follows:

- Initialize all parameters β to some random value
- Repeat until convergence:
 - Choose a new value for β to reduce $\mathcal{E}(\beta)$ toward the direction of steepest descent.
 - Simultaneously update for $j = (0, \dots, d)$

The gradient descent step is formulated by the following function:

$$\beta_j := \beta_j - \alpha \frac{\delta}{\delta \beta_j} \mathcal{E}(\beta)$$

For simple linear regression, let's assume β_0 is included in β_j . We can assume this if we assume $x_0 = 1$. This will keep consistency in our formulas. So, for each j

$$\begin{aligned} \frac{\delta \mathcal{E}}{\delta \beta_j} &= \frac{\delta}{\delta \beta_j} \frac{1}{2n} \sum_{i=1}^n \left(\hat{y}^{(i)} - y^{(i)} \right)^2 \\ &= \frac{\delta}{\delta \beta_j} \frac{1}{2n} \sum_{i=1}^n \left(\sum_{k=0}^d \beta_k + x_k^{(i)} - y^{(i)} \right)^2 \\ &= \frac{1}{n} \sum_{i=1}^n (x_j^{(i)}) \left(\sum_{k=0}^d \beta_k + x_k^{(i)} - y^{(i)} \right) \end{aligned}$$

Thus, for all training examples:

$$\beta_j := \beta_j - \alpha \sum_{i=1}^n x_j^{(i)} \left(\sum_{k=0}^d \beta_k + x_k^{(i)} - y^{(i)} \right)$$

Repeat the above until convergence and update for $j = (0, 1, \dots, d)$ Note that $\mathcal{L}(\beta)$ is a quadratic equation and α is an empirical value. The above equation is also called batch gradient descent. All data is summed at every step, which is very ineffective for big datasets. Alternatively, you can use Stochastic Gradient Descent (SGD):

$$\begin{aligned} &\text{Repeat } \{ \\ &\quad \text{For } i = 1 \text{ to } n \{ \\ &\quad \quad \beta_j := \beta_j - \alpha \left[h_\beta x^{(i)} - y^{(i)} \right] x_j^{(i)} \\ &\quad \quad \} \\ &\quad \} \end{aligned}$$

The above takes derivative of one example at a time. The gradient takes a bit more noisy approach toward minima and it never converges (unlike batch gradient descent).

1.5.3 Probabilistic Solution

1.6 Evaluation

Chapter 2

Logistic Regression