Linear Regression Notes

by Jansen Ken Pegrasio

16 February 2025

1 Introduction

Linear regression works by assuming linear relationship between features and targets. For example, when we assume linear relationship between insurance charges and ages, our machine learning model will look like $charges = w \times age + b$ where w is the weight (slope) and b is the intercept.

In short, the way linear regression works is first by choosing random values for w and b. Using these random values, it will then predict an output. Next, it will calculate the error between the actual and predicted output. This error will then be passed to an optimization method which will reinforce the model.

2 How the Optimization Method Works?

The two most common optimization method are:

1. Gradient Descent

   First, let's assume:

   (a) y_i as the actual output for a given feature x_i

   (b) y_{pi} as the predicted outputs for a given feature x_i

   (c) y_{pi} is predicted by using this formula $y_{pi} = \beta_0 + \beta_1 x_i$, according to the definition of a linear regression model

   Choose sum of squares residual as our loss function! $SSR = \Sigma_{i=1}^{n}(y_i - y_{pi})^2 = \Sigma_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2$. This optimizer works by choosing the weight and the intercept that minimize the residual sum of squares. This means we want to find values of β_0 and β_1 so that the sum of squares residual will be as small as possible. We can find these values by using calculus.

Calculating the derivatives of the loss function with respect to β_0; denote it with $\frac{\partial L}{\partial \beta_0}$:

$$\begin{aligned}
\frac{\partial}{\partial \beta_0}[\Sigma_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2] &= \frac{\partial(\Sigma_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2)}{\partial(y_i - \beta_0 - \beta_1 x_i)} \cdot \frac{\partial(y_i - \beta_0 - \beta_1 x_i)}{\partial \beta_0} \\
&= \Sigma_{i=1}^n 2(y_i - \beta_0 - \beta_1 x_i) \cdot (-1) \\
&= -2 \cdot \Sigma_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)
\end{aligned}$$

Calculating the derivatives of the loss function with respect to β_1; denote it with $\frac{\partial L}{\partial \beta_1}$:

$$\begin{aligned}
\frac{\partial}{\partial \beta_1}[\Sigma_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2] &= \frac{\partial(\Sigma_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2)}{\partial(y_i - \beta_0 - \beta_1 x_i)} \cdot \frac{\partial(y_i - \beta_0 - \beta_1 x_i)}{\partial \beta_1} \\
&= \Sigma_{i=1}^n 2(y_i - \beta_0 - \beta_1 x_i) \cdot (-x_i) \\
&= -2 \cdot \Sigma_{i=1}^n (x_i y_i - x_i \beta_0 - \beta_1 x_i^2)
\end{aligned}$$

Using these information, $\frac{\partial L}{\partial \beta_0}$ and $\frac{\partial L}{\partial \beta_1}$, we will update β_0 and β_1 little by little. Here is the formalized version of it:

(a) $\beta_0 = \beta_0 - \eta \cdot \frac{\partial L}{\partial \beta_0}$

(b) $\beta_1 = \beta_1 - \eta \cdot \frac{\partial L}{\partial \beta_1}$

Note that η is the learning rate!

2. Stochastic Gradient Descent

   Stochastic Gradient Descent is just like Gradient Descent, except it only looks at one mini batch per step. So, instead of calculating the sum of all values x_i and y_i, we only calculate the sum of the values inside the batch.

3 Representing Multiple Linear Regression in Matrix Form

So far, what we have been doing are just Simple Linear Regression where we just use one feature to predict one output. However, in real life cases, that is not usually the case. We will have many features in hand. Therefore, we need a way to generalize our linear regression model, which can be achieved by using matrices!

For every sample i where $1 \leq i \leq n$, let $x_{1i}, x_{2i}, x_{3i}, ..., x_{ki}$ be the features variable to predict y_i. In linear regression, we can represent the predicted value, $\hat{y_i}$ as $\beta_0 + x_{1i}\beta_1 + x_{2i}\beta_2 + ... + x_{ki}\beta_j$ where $\beta_0, \beta_1, \beta_2, ..., \beta_k$ represent the weights assigned for each feature $x_{1i}, x_{2i}, x_{3i}, ..., x_{ki}$.

Other way to represent this simpler is by using matrices. First, let's group all actual predictions in a single column ($n \times 1$) matrix, $\boldsymbol{y}$.

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ ... \\ y_n \end{bmatrix}$$

Next, we define a **design matrix**, $\boldsymbol{x}$ to represent our features variable. Notice that we add ones in the leftmost column for the biases to fill in.

$$\boldsymbol{x} = \begin{bmatrix} 1 & x_{11} & x_{21} & ... & x_{k1} \\ 1 & x_{12} & x_{22} & ... & x_{k2} \\ 1 & x_{13} & x_{23} & ... & x_{k3} \\ ... & ... & ... & ... & ... \\ 1 & x_{1n} & x_{2n} & ... & x_{kn} \end{bmatrix}$$

Last, we also design a matrix to represent the weights corresponding to each feature, call this β.

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ ... \\ \beta_k \end{bmatrix}$$

We represent our model as $\hat{y} = \boldsymbol{x}\beta$!

Let's define $e(\beta) = \boldsymbol{y} - \boldsymbol{x}\beta$ as our residual $(k + 1) \times 1$ matrix for each corresponding β_i value. Using this information, let's represent our sum of squared residuals loss function. To help, let's observe this matrix multiplication:

$$\begin{bmatrix} e_0 & e_1 & e_2 & ... & e_k \end{bmatrix} \begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ ... \\ e_k \end{bmatrix}$$

Indeed, this matrix multiplication is what we want to represent in our sum of squared residuals function. Let's describe it! Let $SSR(\beta)$ be the loss function where each row corresponds to each β_i.

$$SSR(\beta) = e^T e$$

Substituting $e(\beta) = \boldsymbol{y} - \boldsymbol{x}\beta$, we can get:

$$\begin{aligned} SSR(\beta) &= (\boldsymbol{y} - \boldsymbol{x}\beta)^T \cdot (\boldsymbol{y} - \boldsymbol{x}\beta) \\ SSR(\beta) &= (\boldsymbol{y}^T - \beta^T \boldsymbol{x}^T) \cdot (\boldsymbol{y} - \boldsymbol{x}\beta) \\ SSR(\beta) &= (\boldsymbol{y}^T y - \boldsymbol{y}^T \boldsymbol{x}\beta - \beta^T \boldsymbol{x}^T \boldsymbol{y} + \beta^T \boldsymbol{x}^T \boldsymbol{x}\beta) \end{aligned}$$

Notice that $\boldsymbol{y}^T x\beta = (\beta^T \boldsymbol{x}^T \boldsymbol{y})^T = \beta^T \boldsymbol{x}^T \boldsymbol{y}$ because $\beta^T \boldsymbol{x}^T \boldsymbol{y}$ is a 1×1 matrix. Thus, we can further simplify our $SSR(\beta)$ function!

$$SSR(\beta) = \boldsymbol{y}^T y - 2\boldsymbol{y}^T \boldsymbol{x}\beta + \beta^T \boldsymbol{x}^T \boldsymbol{x}\beta$$

4 Gradient in Higher Dimension Properties and Proofs

Before we differentiate our loss function, we need to understand some properties of gradient in higher dimension. There are two properties that we can achieve:

1. Linear Function: $\nabla_x(a^T x + b) = a$

   Assume a and x are column vectors with size of n. Let's represent $a^T x$ as its summation notation form.

   $$a^T x = \begin{bmatrix} a_1 & a_2 & a_3 & ... & a_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ ... \\ x_n \end{bmatrix} = \Sigma_{i=1}^n (a_i \cdot x_i)$$

   Now, let's find the gradient of $a^T x$ with respect to x! Finding gradient means: for $1 \leq k \leq n$, we will take the derivatives of $\Sigma_{i=1}^n (a_i \cdot x_i)$ with respect to x_k!

   $$\begin{aligned} \frac{\partial}{\partial x_k} \Sigma_{i=1}^n (a_i \cdot x_i) &= \frac{\partial}{\partial x_k}(\Sigma_{i \neq k} a_i x_i + a_k x_k) = \frac{\partial}{\partial x_k}(\Sigma_{i \neq k} a_i x_i) + \frac{\partial}{\partial x_k} a_k x_k \\ &= 0 + a_k = a_k \end{aligned}$$

   Now, with this derivatives, we can assemble the partial derivatives into vector $\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ ... \\ a_n \end{bmatrix} = a$. Thus, using this information, we can conclude that $\nabla_x(a^T x + b) = \nabla_x(a^T x) + \nabla_x(b) = a + 0 = a$

2. Quadratic Function: $\nabla_x(x^T A x) = 2Ax$

Assume A is a $n \times n$ matrix, and x is a column matrix of size n.

$$\begin{aligned}
x^T A x &= x^T \cdot \begin{bmatrix} a_{11} & a_{12} & a_{13} & ... & a_{1n} \\ a_{21} & a_{22} & a_{23} & ... & a_{2n} \\ a_{31} & a_{32} & a_{33} & ... & a_{3n} \\ ... & ... & ... & ... & ... \\ a_{n1} & a_{n2} & a_{n3} & ... & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ ... \\ x_n \end{bmatrix} \\
&= x^T \cdot \begin{bmatrix} \Sigma_{i=1}^n a_{1i}x_i \\ \Sigma_{i=1}^n a_{2i}x_i \\ \Sigma_{i=1}^n a_{3i}x_i \\ ... \\ \Sigma_{i=1}^n a_{ni}x_i \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & ... & x_n \end{bmatrix} \begin{bmatrix} \Sigma_{i=1}^n a_{1i}x_i \\ \Sigma_{i=1}^n a_{2i}x_i \\ \Sigma_{i=1}^n a_{3i}x_i \\ ... \\ \Sigma_{i=1}^n a_{ni}x_i \end{bmatrix} \\
&= \Sigma_{j=1}^n (x_j \cdot \Sigma_{i=1}^n a_{ji}x_i) = \Sigma_{j=1}^n \Sigma_{i=1}^n x_j a_{ji} x_i
\end{aligned}$$

Similar to before, let's take the derivatives of $\Sigma_{j=1}^n \Sigma_{i=1}^n x_j a_{ji} x_i$ with respect to x_k.

(a) Consider the case when $j = k$ and $i = k$:

$$\frac{\partial}{\partial x_k} \Sigma_{j=k} \Sigma_{i=k} x_j a_{ji} x_i = \frac{\partial}{\partial x_k} x_k a_{kk} x_k = \frac{\partial}{\partial x_k} a_{kk} x_k^2 = 2a_{kk}x_k$$

(b) Consider the case when $j = k$ and $i \neq k$:

$$\begin{aligned}
\frac{\partial}{\partial x_k} \Sigma_{j=k} \Sigma_{i\neq k} x_j a_{ji} x_i &= \frac{\partial}{\partial x_k} \Sigma_{i\neq k} x_k a_{ki} x_i = \Sigma_{i\neq k} (\frac{\partial}{\partial x_k} x_k a_{ki} x_i) \\
&= \Sigma_{i\neq k} a_{ki} x_i
\end{aligned}$$

(c) Consider the case when $j \neq k$ and $i = k$:

$$\begin{aligned}
\frac{\partial}{\partial x_k} \Sigma_{j\neq k} \Sigma_{i=k} x_j a_{ji} x_i &= \frac{\partial}{\partial x_k} \Sigma_{j\neq k} x_j a_{jk} x_k = \Sigma_{j\neq k} (\frac{\partial}{\partial x_k} x_j a_{jk} x_k) \\
&= \Sigma_{j\neq k} x_j a_{jk}
\end{aligned}$$

(d) Consider the case when $j \neq k$ and $i \neq k$:

$$\frac{\partial}{\partial x_k} \Sigma_{j\neq k} \Sigma_{i\neq k} x_j a_{ji} x_i = 0$$

Combining these cases together, we would have:

$$\begin{aligned}
\frac{\partial}{\partial x_k} \Sigma_{j=1}^n \Sigma_{i=1}^n x_j a_{ji} x_i &= 2a_{kk}x_k + \Sigma_{i\neq k} a_{ki} x_i + \Sigma_{j\neq k} x_j a_{jk} \\
&= (a_{kk}x_k + \Sigma_{i\neq k} a_{ki} x_i + (a_{kk}x_k + \Sigma_{j\neq k} x_j a_{jk}) \\
&= (\Sigma_{i=k} a_{ki} x_i + \Sigma_{i\neq k} a_{ki} x_i) + (\Sigma_{j=k} x_j a_{jk} + \Sigma_{j\neq k} x_j a_{jk}) \\
&= \Sigma_{i=1}^n a_{ki} x_i + \Sigma_{j=1}^n x_j a_{jk}
\end{aligned}$$

Let's assemble these partial derivatives into:

$$\begin{bmatrix} \Sigma_{i=1}^{n} a_{1i}x_i + \Sigma_{j=1}^{n} x_j a_{j1} \\ \Sigma_{i=1}^{n} a_{2i}x_i + \Sigma_{j=1}^{n} x_j a_{j2} \\ \Sigma_{i=1}^{n} a_{3i}x_i + \Sigma_{j=1}^{n} x_j a_{j3} \\ ... \\ \Sigma_{i=1}^{n} a_{ni}x_i + \Sigma_{j=1}^{n} x_j a_{jn} \end{bmatrix} = \begin{bmatrix} \Sigma_{i=1}^{n} a_{1i}x_i \\ \Sigma_{i=1}^{n} a_{2i}x_i \\ \Sigma_{i=1}^{n} a_{3i}x_i \\ ... \\ \Sigma_{i=1}^{n} a_{ni}x_i \end{bmatrix} + \begin{bmatrix} \Sigma_{j=1}^{n} a_{j1}x_j \\ \Sigma_{j=1}^{n} a_{j2}x_j \\ \Sigma_{j=1}^{n} a_{j3}x_j \\ ... \\ \Sigma_{j=1}^{n} x_j a_{jn} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ a_{31} & a_{32} & ... & a_{3n} \\ ... & & & \\ a_{n1} & a_{n2} & ... & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ ... \\ x_n \end{bmatrix} + \begin{bmatrix} a_{11} & a_{21} & ... & a_{n1} \\ a_{12} & a_{22} & ... & a_{n2} \\ a_{13} & a_{23} & ... & a_{n3} \\ ... & & & \\ a_{1n} & a_{2n} & ... & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ ... \\ x_n \end{bmatrix}$$

$$= Ax + A^T x = (A + A^T)x$$

Special condition occurs when $A = A^T$; it will make $\frac{\partial}{\partial x}(x^T Ax) = 2Ax$. Otherwise, $\frac{\partial}{\partial x}(x^T Ax) = (A + A^T)x$

5 Gradient Descent Applied to the Matrix Form of Linear Regression

Now, let's use what we have proved in Chapter 4 to differentiate our $SSR(\beta)$ function in Chapter 3.

$$\begin{aligned} \nabla_\beta SSR(\beta) &= \nabla_\beta \boldsymbol{y}^T \boldsymbol{y} - 2\nabla_\beta \boldsymbol{y}^T \boldsymbol{x}\beta + \nabla_\beta \beta^T \boldsymbol{x}^T \boldsymbol{x}\beta \\ &= 0 - 2(\boldsymbol{y}^T \boldsymbol{x})^T + \nabla_\beta \beta^T \boldsymbol{x}^T \boldsymbol{x}\beta && \text{(according to Chapter 3.1)} \\ &= -2\boldsymbol{x}^T \boldsymbol{y} + \nabla_\beta \beta^T \boldsymbol{x}^T \boldsymbol{x}\beta && \text{(transpose property)} \\ &= -2\boldsymbol{x}^T \boldsymbol{y} + 2\boldsymbol{x}^T \boldsymbol{x}\beta && (\boldsymbol{x}^T \boldsymbol{x} \text{ is symmetric, apply Chapter 3.2}) \\ &= -2(\boldsymbol{x}^T \boldsymbol{y} - \boldsymbol{x}^T \boldsymbol{x}\beta) \\ &= -2\boldsymbol{x}^T (\boldsymbol{y} - \boldsymbol{x}\beta) \end{aligned}$$

Notice when we are calculating for β_0, the value of x^T is just a row of ones. Thus, for β_0, we can say that $\nabla_\beta SSR(\beta) = -2(\boldsymbol{y} - \boldsymbol{x}\beta)$.

6 Implementation Details

When implementing the idea of linear regression, we can choose to separate weight and intercept to two different variables; name them *intercept* and *weight*. *intercept* will just be filled by β_0, whereas *weight* is a column vector corresponds to $\beta_1, \beta_2, ..., \beta_k$.

For calculating the rate of change of β_0, we can use this formula $-2 \cdot error$ where $error = actual - prediction$. Then, for calculating the rate of change of β_1, we can use the formula: $-2\boldsymbol{x}^T \cdot error$.

7 Reference

1. Lecture 13 - Statistical Models (CMU)
2. Linear Quadratic Gradients (UBC)