Derivation of Linear Regression

Author: Sami Abu-El-Haija (samihaija@umich.edu)

We derive, step-by-step, the Linear Regression Algorithm, using Matrix Algebra. Linear Regression is generally used to predict a continuous value. For example, predicting the price of a house. Linear Regression algorithms process a dataset of the form $\{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_N, t_N)\}$. Where $\mathbf{x}_n$ and t_n are, respectively, the features and the true/target value of the n-th training example.

1 Matrix Calculus Formulae

During this derivation, we will assume that you are familiar with deriving the following matrix calculus rules:

- $\nabla_{\mathbf{x}} b^T \mathbf{x} = b$
- $\nabla_{\mathbf{x}} \mathbf{x}^T A \mathbf{x} = 2A\mathbf{x} \quad (if A \in \mathbb{S})$
- $\nabla^2_{\mathbf{x}} \mathbf{x}^T A \mathbf{x} = 2A \quad (if A \in \mathbb{S})$

If you are not familiar with these rules, please study (and **derive**) sections 4.3 and 4.4 of `http://cs229.stanford.edu/section/cs229-linalg.pdf`

2 Setting up system of equations

2.1 Reminder: from Algebra to Matrices

Generally, if you have a system of equations:

$$w_1 \times 14 + w_2 \times 16 = 7$$

$$w_1 \times 4 + w_2 \times 4 = 13$$

Where you would like to find the w_1 and w_2 that satisfy the above equations, you could solve for w_1 from the first equation and plug it into the second. Or alternatively, you can setup a Matrix multiplication that is equivalent to the above equations as:

$$\begin{bmatrix} 14 & 16 \\ 4 & 4 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 7 \\ 13 \end{bmatrix}$$

You can then solve for w_1 and w_2 by inverting the left-matrix and multiplying it by the right-hand side.

2.2 Linear Regression in Matrix form

Assume your training data is *housing prices*. Each house has its features: square-foot area, number of bedrooms, number of bathrooms, ... Each house has a price. The data can be written this tabular form:

$\mathbf{x} =$	[area	bedrooms	bathrooms]	t
	2000	3	2	\$200,000
	2500	4	4	\$280,000
	1300	1	1	\$130,000
	⋮			⋮

Generally speaking, we can represent our data as a *data matrix* and a *target values vector* as:

$$\begin{bmatrix} \text{———} & \mathbf{x}_1^T & \text{———} \\ \text{———} & \mathbf{x}_2^T & \text{———} \\ & \vdots & \\ & \vdots & \\ \text{———} & \mathbf{x}_N^T & \text{———} \end{bmatrix} \text{ and } \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ \vdots \\ t_N \end{bmatrix}$$

In addition, it is often times desirable to have a *transformation* of the features of training examples. This is done through a *basis function* (denoted $\phi(.)$). Just for the sake of demonstration, if $\mathbf{x} = [x_1, x_2, x_3]$ where x_1, x_2, x_3 are [square-foot area, # bedrooms, # bathrooms], then one could design a basis function of $\phi(\mathbf{x}) = [1, log(x_1), x_2^2, x_3, x_3^2]$. Let the size of the vector produced by $\phi(.)$ be M. Normally, basis functions are designed around intuitions around data. Each application (Information Retrieval versus Computer Vision) have different ways in coming-up with basis functions that fits the field of interest best. Nonetheless, our data can be represented as a *design matrix* and a *target values vector* as:

$$\begin{bmatrix} \text{———} & \phi(\mathbf{x}_1)^T & \text{———} \\ \text{———} & \phi(\mathbf{x}_2)^T & \text{———} \\ & \vdots & \\ & \vdots & \\ \text{———} & \phi(\mathbf{x}_N)^T & \text{———} \end{bmatrix} \text{ and } \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ \vdots \\ t_N \end{bmatrix}$$

The Matrix Formulation of linear regression try to find the weights vector $\mathbf{w} = [\mathbf{w_1}, \ldots, \mathbf{w_M}]$ which satisfies:

$$\begin{bmatrix} \text{———} & \phi(\mathbf{x}_1)^T & \text{———} \\ \text{———} & \phi(\mathbf{x}_2)^T & \text{———} \\ & \vdots & \\ & \vdots & \\ \text{———} & \phi(\mathbf{x}_N)^T & \text{———} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ \vdots \\ t_N \end{bmatrix}$$

Where $\phi(\mathbf{x}) \in \mathbb{R}^M$. In practice, $M << N$, because of the abundance of training data and because M should not be chosen larger than N to avoid over-fitting. In the usual case of $M << N$, there aren't a set of weights $w_1, \ldots, w_M$ that exactly solve the above equation (for example, the design matrix is not invertible, or as in most cases, not a square matrix). Therefore, the linear regression algorithm tries to find the weights $\mathbf{w}$ that produce values that are *as close as possible* to the (true) target values. Notice that the left-hand side in the equation above ($\Phi\mathbf{w}$) is a vector of size N. Informally, we want to find the $\mathbf{w}$ that:

$$\begin{bmatrix} \text{———} & \phi(\mathbf{x}_1)^T & \text{———} \\ \text{———} & \phi(\mathbf{x}_2)^T & \text{———} \\ & \vdots & \\ & \vdots & \\ \text{———} & \phi(\mathbf{x}_N)^T & \text{———} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix} \approx \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ \vdots \\ t_N \end{bmatrix}$$

Or, to reduce notation:

$$\Phi\mathbf{w} \approx \mathbf{t}$$

3 Solving for w in Matrix form

The $\mathbf{w}$ that makes the left-hand side as close as possible to the right-hand side can be written as this minimization problem:

$$\min_{\mathbf{w}} \frac{1}{2} ||\Phi\mathbf{w} - \mathbf{t}||^{\mathbf{2}}$$

The term under the minimization is referred to *squared error* (also known as *sum of squares error*). It is common to write the above as:

$$\min_{\mathbf{w}} E(\mathbf{w}); \text{ where } \mathbf{E}(\mathbf{w}) = \frac{\mathbf{1}}{\mathbf{2}}||\mathbf{\Phi w} - \mathbf{t}||^{\mathbf{2}}$$

Recall that $||\mathbf{v}||$ is the norm (L2-norm) of a vector $\mathbf{v}$. Also recall the recall the identity

$$||\mathbf{v}|| = \sqrt{\sum_{\mathbf{i}}^{\mathbf{N}} \mathbf{v}_{\mathbf{i}}^{\mathbf{2}}} = \sqrt{\mathbf{v}^{\mathbf{T}}\mathbf{v}}; \text{ where} \mathbf{v} \in \mathbb{R}^{\mathbf{N}}$$

Consequently:

$$\min_{\mathbf{w}} \frac{1}{2}||\Phi\mathbf{w} - \mathbf{t}||^{\mathbf{2}} = \min_{\mathbf{w}}(\mathbf{\Phi w} - \mathbf{t})^{\mathbf{T}}(\mathbf{\Phi w} - \mathbf{t})$$

To minimize the above expression with respect to $\mathbf{w}$, we can take the (vector) derivative of the expression with respect to $\mathbf{w}$ as:

$$\nabla_{\mathbf{w}}\left[\frac{1}{2}(\Phi\mathbf{w} - \mathbf{t})^{\mathbf{T}}(\mathbf{\Phi w} - \mathbf{t})\right] = \nabla_{\mathbf{w}}\left[\frac{1}{2}(\mathbf{w}^{\mathbf{T}}\mathbf{\Phi}^{\mathbf{T}} - \mathbf{t}^{\mathbf{T}})(\mathbf{\Phi w} - \mathbf{t})\right]$$

$$= \nabla_{\mathbf{w}}\left[\frac{1}{2}(\mathbf{w}^{\mathbf{T}}\mathbf{\Phi}^{\mathbf{T}}\mathbf{\Phi w} - \mathbf{t}^{\mathbf{T}}\mathbf{\Phi w} - \mathbf{w}^{\mathbf{T}}\mathbf{\Phi}^{\mathbf{T}}\mathbf{t} + \mathbf{t}^{\mathbf{T}}\mathbf{t})\right]$$

Note that the last term in the above expression does not depend on $\mathbf{w}$ and its derivative w.r.t $\mathbf{w}$ is zero. In addition, $\mathbf{w}^{\mathbf{T}}\mathbf{\Phi}^{\mathbf{T}}\mathbf{t}$ (and also $\mathbf{t}^T\Phi\mathbf{w}$) is a 1×1 matrix (a real number). Work-out the dimensions of this to verify yourself. The transpose of a real-number is itself. i.e. $\mathbf{w}^{\mathbf{T}}\mathbf{\Phi}^{\mathbf{T}}\mathbf{t} = (\mathbf{w}^{\mathbf{T}}\mathbf{\Phi}^{\mathbf{T}}\mathbf{t})^{\mathbf{T}} = \mathbf{t}^{\mathbf{T}}\mathbf{\Phi w}$. Therefore, the above expression simplifies to:

$$= \nabla_{\mathbf{w}}\left[\frac{1}{2}(\mathbf{w}^{\mathbf{T}}\mathbf{\Phi}^{\mathbf{T}}\mathbf{\Phi w} - \mathbf{2t}^{\mathbf{T}}\mathbf{\Phi w})\right]$$

Using the matrix calculus formulae at the beginning of this document, the above derivative is:

$$\nabla_{\mathbf{w}}E(\mathbf{w}) = \frac{\mathbf{1}}{\mathbf{2}}(\mathbf{2\Phi}^{\mathbf{T}}\mathbf{\Phi w} - \mathbf{2}(\mathbf{t}^{\mathbf{T}}\mathbf{\Phi})^{\mathbf{T}}) = \mathbf{\Phi}^{\mathbf{T}}\mathbf{\Phi w} - \mathbf{\Phi}^{\mathbf{T}}\mathbf{t}$$

Note that $\Phi^T\Phi$ is symmetric.

3.1 Closed Form Solution

The $\mathbf{w}$ that minimizes $E(\mathbf{w})$ (lets call it $\mathbf{w}^*$ can be found in multiple ways. Solving for the minimum $\mathbf{w}^*$ in *closed form*, means that solving for $\mathbf{w}$ in $\nabla_{\mathbf{w}}E(\mathbf{w}) = \mathbf{0}$:

$$\Phi^T\Phi\mathbf{w}^* - \Phi^T\mathbf{t} = 0$$

$$\Phi^T\Phi\mathbf{w}^* = \Phi^T\mathbf{t}$$

$$\mathbf{w}^* = (\Phi^T\Phi)^{-1}\Phi^T\mathbf{t}$$

3.2 Gradient Descent

Gradient Descent and Stochastic Gradient Descent are not going to be in this handout for now. They will be added later

4 Regularized Linear Regression

In practice, it works *better* to fit the **w** parameter to the data while also placing a constraint on **w** to be small. This makes the **w** generalize better and have better performance on unseen test data. It is common to do set up the following minimization problem:

$$\begin{aligned}
\min_{\mathbf{w}} \tilde{E}(\mathbf{w}) &= \frac{1}{2}||\Phi\mathbf{w} - \mathbf{t}||^{\mathbf{2}} + \frac{\lambda}{\mathbf{2}}||\mathbf{w}|| \\
\nabla_{\mathbf{w}} \tilde{E}(\mathbf{w}) &= \nabla_{\mathbf{w}} \left[\frac{1}{2}(\Phi\mathbf{w} - \mathbf{t})^{\mathbf{T}}(\boldsymbol{\Phi}\mathbf{w} - \mathbf{t}) + \frac{\lambda}{\mathbf{2}}\mathbf{w}^{\mathbf{T}}\mathbf{w}\right] \\
&= \nabla_{\mathbf{w}} \left[\frac{1}{2}(\Phi\mathbf{w} - \mathbf{t})^{\mathbf{T}}(\boldsymbol{\Phi}\mathbf{w} - \mathbf{t}) + \frac{\lambda}{\mathbf{2}}\mathbf{w}^{\mathbf{T}}\mathbf{I}\mathbf{w}\right] \\
&= \Phi^T\Phi\mathbf{w} - \boldsymbol{\Phi}^{\mathbf{T}}\mathbf{t} + \lambda\mathbf{w}
\end{aligned}$$

In closed form:

$$\begin{gathered}
\Phi^T\Phi\mathbf{w}^* - \Phi^T\mathbf{t} + \lambda\mathbf{w}^* = 0 \\
\Phi^T\Phi\mathbf{w}^* + \lambda\mathbf{w}^* = \Phi^T\mathbf{t} \\
(\Phi^T\Phi + \lambda\mathbf{I})\mathbf{w}^* = \boldsymbol{\Phi}^{\mathbf{T}}\mathbf{t} \\
\mathbf{w}^* = (\Phi^T\Phi + \lambda\mathbf{I})^{\mathbf{-1}}\boldsymbol{\Phi}^{\mathbf{T}}\mathbf{t}
\end{gathered}$$