The Normal Equation and matrix calculus (http://eli.thegreenplace.net/2015/the-normal-equation-and-matrix-calculus/)

May 27, 2015 at 06:19 **Tags** Math (http://eli.thegreenplace.net/tag/math) , Machine Learning (http://eli.thegreenplace.net/tag/machine-learning)

A few months ago I wrote a post (http://eli.thegreenplace.net/2014/derivation-of-the-normal-equation-for-linear-regression) on formulating the Normal Equation for linear regression. A crucial part in the formulation is using matrix calculus (http://en.wikipedia.org/wiki/Matrix_calculus) to compute a scalar-by-vector derivative. I didn't spend much time explaining how this step works, instead remarking:

> Deriving by a vector may feel uncomfortable, but there's nothing to worry about. Recall that here we only use matrix notation to conveniently represent a system of linear formulae. So we derive by each component of the vector, and then combine the resulting derivatives into a vector again.

According to the comments received on the post, folks didn't find this convincing and asked for more details. One commenter even said that "matrix calculus feels handwavy", something which I fully agree with. The reason matrix calculus feels handwavy is that it's not as commonly encountered as "regular" calculus, and hence its identities and intuitions are not as familiar. However, there's really not that much to it, as I want to show here.

Let's get started with a simple example, which I'll use to demonstrate the principles. Say we have the function:

$$f(v) = a^T v$$

Where **a** and **v** are vectors with *n* components [1]. We want to compute its derivative by **v**. But wait, while a "regular" derivative by a scalar is clearly defined (using limits), what does deriving by a vector mean? It simply means that we derive by each component of the vector separately, and then combine the results into a new vector [2]. In other words:

$$\begin{pmatrix} \frac{\partial f}{\partial v_1} \end{pmatrix}$$

$$\frac{\partial f}{\partial v} = \begin{pmatrix} \\ \frac{\partial f}{\partial v_2} \\ \dots \\ \frac{\partial f}{\partial v_n} \\ \end{pmatrix}$$

Let's see how this works out for our function *f*. It may be more convenient to rewrite it by using components rather than vector notation:

$$f(v) = a^T v = a_1 v_1 + a_2 v_2 + \dots + a_n v_n$$

Computing the derivatives by each component, we'll get:

$$\frac{\partial f}{\partial v_1} = a_1$$

$$\frac{\partial f}{\partial v_2} = a_2$$

$$\dots$$

$$\frac{\partial f}{\partial v_n} = a_n$$

So we have a sequence of partial derivatives, which we combine into a vector:

$$\frac{\partial f}{\partial v} = \begin{pmatrix} a_1 \\ \dots \\ a_n \end{pmatrix}$$

Or, in other words $\frac{\partial f}{\partial v} = a$.

This example demonstrates the algorithm for computing scalar-by-vector derivatives:

1. Figure out what the dimensions of all vectors and matrices are.
2. Expand the vector equations into their full form (a multiplication of two vectors is either a scalar or a matrix, depending on their orientation, etc.) Note that this will end up with a scalar.
3. Compute the derivative of the scalar by each component of the variable vector separately.
4. Combine the derivatives into a vector.

Similarly to regular calculus, matrix and vector calculus rely on a set of identities to make computations more manageable. We can either go the hard way (computing the derivative of each function from basic principles using limits), or the easy way - applying the plethora of convenient identities that were developed to make this task simpler. The identity for computing the derivative of $a^T v$ shown above plays the role of $\frac{d}{dx} ax = a$ in regular calculus.

Now we have the tools to understand how the vector derivatives in the normal equation article (http://eli.thegreenplace.net/2014/derivation-of-the-normal-equation-for-linear-regression) were computed. As a reminder, this is the matrix form of the cost function *J*:

$$J(\theta) = \theta^T X^T X \theta - 2(X\theta)^T y + y^T y$$

And we're interested in computing $\frac{\partial J}{\partial \theta}$. The equation for *J* consists of three terms added together. The last one $y^T y$ doesn't contribute to the derivative because it doesn't depend on the variable. Let's start looking at the second (since it's simpler than the first) - and give it a name, for convenience:

$$P(\theta) = 2(X\theta)^T y$$

We'll start by recalling what all the dimensions are. θ is a vector of n components. y is a vector of m components. X is a m-by-n matrix.

Let's see what *P* expands to [3]:

$$P(\theta) = 2 \left[\begin{pmatrix} x_{11} & x_{12} & ... & x_{1n} \\ x_{21} & ... & ... & x_{2n} \\ ... & & & \\ x_{m1} & ... & ... & x_{mn} \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \\ ... \\ \theta_n \end{pmatrix} \right]^T \begin{pmatrix} y_1 \\ y_2 \\ ... \\ y_m \end{pmatrix}$$

Computing the matrix-by-vector multiplication inside the parens:

$$P(x) = 2 \left[\begin{pmatrix} x_{11}\theta_1 + ... + x_{1n}\theta_n \\ x_{21}\theta_1 + ... + x_{2n}\theta_n \\ ... \\ x_{m1}\theta_1 + ... + x_{mn}\theta_n \end{pmatrix} \right]^T \begin{pmatrix} y_1 \\ y_2 \\ ... \\ y_m \end{pmatrix}$$

And finally, multiplying the two vectors together:

$$P(x) = 2(x_{11}\theta_1 + ... + x_{1n}\theta_n)y_1 + 2(x_{21}\theta_1 + ... + x_{2n}\theta_n)y_2 + ... + 2(x_{m1}\theta_1 + ... + x_{mn}\theta_n)y_m$$

Working with such formulae makes you appreciate why mathematicians have long ago come up with shorthand notations like "sigma" summation:

$$P(x) = 2\sum_{r=1}^{m} y_r(x_{r1}\theta_1 + ... + x_{rn}\theta_n) = 2\sum_{r=1}^{m} y_r \sum_{c=1}^{n} x_{rc}\theta_c$$

OK, so we've finally completed step 2 of the algorithm - we have the scalar equation for *P*. Now it's time to compute its derivative by each θ:

$$\begin{aligned} \frac{\partial P}{\partial \theta_1} &= 2(x_{11}y_1 + ... + x_{m1}y_m) \\ \frac{\partial P}{\partial \theta_2} &= 2(x_{12}y_1 + ... + x_{m2}y_m) \\ &... \\ \frac{\partial P}{\partial \theta_n} &= 2(x_{1n}y_1 + ... + x_{mn}y_m) \end{aligned}$$

Now comes the most interesting part. If we treat $\frac{\partial P}{\partial \theta}$ as a vector of n components, we can rewrite this set of equations using a matrix-by-vector multiplication:

$$\frac{\partial P}{\partial \theta} = 2X^T y$$

Take a moment to convince yourself this is true. It's just collecting the individual components of **X** into a matrix and the individual components of **y** into a vector. Since **X** is a m-by-n matrix and **y** is a m-by-1 column vector, the dimensions work out and the result is a n-by-1 column vector.

So we've just computed the second term of the vector derivative of *J*. In the process, we've discovered a useful vector derivative identity for a matrix **X** and vectors θ and **y**:

$$\frac{\partial (X\theta)^T y}{\partial \theta} = X^T y$$

OK, now let's get back to the full definition of *J* and see how to compute the derivative of its first term. We'll give it the name *Q*:

$$Q(\theta) = \theta^T X^T X \theta$$

This derivation is somewhat more complex, since θ appears twice in the equation. Here's the equation again with all the matrices and vectors fully laid out (note that I've already done the transposes):

$$Q(\theta) = (\theta_1 ... \theta_n) \begin{pmatrix} x_{11} & x_{21} & ... & x_{m1} \\ x_{12} & ... & ... & x_{m2} \\ ... & & & \\ x_{1n} & ... & ... & x_{mn} \end{pmatrix} \begin{pmatrix} x_{11} & x_{12} & ... & x_{1n} \\ x_{21} & ... & ... & x_{2n} \\ ... & & & \\ x_{m1} & ... & ... & x_{mn} \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \\ ... \\ \theta_n \end{pmatrix}$$

I'll just multiply the two matrices in the middle together. The result is a "**X** squared" matrix, which is n-by-n. The element in row *r* and column *c* of this square matrix is:

$$\sum_{i=1}^{m} x_{ir} x_{ic}$$

Note that "**X** squared" is a symmetric matrix (this fact will be important later on). For simplicity of notation, we'll call its elements X^2_{rc}. Multiplying by the θ vector on the right we get:

$$Q(\theta) = (\theta_1 ... \theta_n) \begin{pmatrix} X^2_{11}\theta_1 + ... + X^2_{1n}\theta_n \\ X^2_{21}\theta_1 + ... + X^2_{2n}\theta_n \\ ... \\ X^2_{n1}\theta_1 + ... + X^2_{nn}\theta_n \end{pmatrix}$$

And left-multiplying by θ to get the fully unwrapped formula for *Q*:

$$Q(\theta) = \theta_1(X^2_{11}\theta_1 + ... + X^2_{1n}\theta_n) + \theta_2(X^2_{21}\theta_1 + ... + X^2_{2n}\theta_n) + ... + \theta_n(X^2_{n1}\theta_1 + ... + X^2_{nn}\theta_n)$$

Once again, it's now time to compute the derivatives. Let's focus on $\frac{\partial Q}{\partial \theta_1}$, from which we can infer the rest:

$$\frac{\partial Q}{\partial \theta_1} = (2\theta_1 X^2_{11} + \theta_2 X^2_{12} + ... + \theta_n X^2_{1n}) + \theta_2 X^2_{21} + ... + \theta_n X^2_{n1}$$

Using the fact that **X** squared is symmetric, we know that $X^2_{12} = X^2_{21}$ and so on. Therefore:

$$\frac{\partial Q}{\partial \theta_1} = 2\theta_1 X_{11}^2 + 2\theta_2 X_{12}^2 + ... + 2\theta_n X_{1n}^2$$

The partial derivatives by other θ components are similar. Collecting the sequence of partial derivatives back into a vector equation, we get:

$$\frac{\partial Q}{\partial \theta} = 2X^2\theta = 2X^T X\theta$$

Now back to *J*. Recall that for convenience we broke *J* up into three parts: *P*, *Q* and $y^T y$; the latter doesn't depend on θ so it doesn't play a role in the derivative. Collecting our results from this post, we then get:

$$\frac{\partial J}{\partial \theta} = \frac{\partial Q}{\partial \theta} - \frac{\partial P}{\partial \theta} = 2X^T X\theta - 2X^T y$$

Which is exactly the equation we were expecting to see.

To conclude - if matrix calculus feels handwavy, it's because its identities are less familiar. In a sense, it's handwavy in the same way $\frac{dx^2}{dx} = 2x$ is handwavy. We remember the identity so we don't have to recalculate it every time from first principles. Once you get some experience with matrix calculus, parts of equations start looking familiar and you no longer need to engage in the long and tiresome computations demonstrated here. It's perfectly fine to just remember that the derivative of $\theta^T X\theta$ with a symmetric **X** is $2X\theta$. See the "identities" section of the wikipedia article on matrix calculus (http://en.wikipedia.org/wiki/Matrix_calculus) for many more examples.

---

[1] A few words on notation: by default, a vector **v** is a *column* vector. To get its row version, we transpose it. Moreover, in the vector derivative equations that follow I'm using denominator layout notation (http://en.wikipedia.org/wiki/Matrix_calculus#Layout_conventions). This is not super-important though; as the Wikipedia article suggests, many mathematical papers and writings aren't consistent about this and it's perfectly possible to understand the derivations regardless.

[2] Yes, this is exactly like computing a gradient of a multivariate function.

[3] Take a minute to convince yourself that the dimensions of this equation work out and the result is a scalar.

Comments

14 Comments **Eli Bendersky's website** **1 Kil Kal**

Recommend 3 **Share** **Sort by Oldest**

Join the discussion…

sgsfak • 2 years ago

Tom Minka has some tricks for differentiation here: http://research.microsoft.com/... where the differentials are computed based on their "linear approximation" properties. This technique is further (and initially?) described in Magnus and Neudecker "Matrix differential calculus with applications in statistics and econometrics" (1988).

Reply • Share ›

Matej Briškár • a year ago

I think there is a small typo in the first part of the right side of equation http://eli.thegreenplace.net/i.... You are lacking thetas there. Otherwise it is OK and thank you for this post. I have never studied derivation with matrices, therefore these posts are really helpful for me..

1 Reply • Share ›

Eli Bendersky Mod ➚ Matej Briškár • a year ago

Fixed, thanks.

Reply • Share ›

Santosh Banerjee • a year ago

Pretty awesome post! Really appreciate the explanation of the derivation process involving matrices with elaborate expansions. Was wondering how to extend the max-min theorem in univariate calculus to the case of differentiating a scalar wrt a column vector. In other words, since we are solving for theta in the equation del(J)/del(theta) = 0, pretty much like solving f'(x) = 0, the implicit assumption is that J attains minimum where all the partial derivatives disappear. How do you go about proving that part?

Reply • Share ›

Tom Wilson • 10 months ago

Thank you for taking the time to go through this derivation - it's very much appreciated. Thank you!

Reply • Share ›

Mike • 9 months ago

↑ Back to top