Notes on Andrew Ng's CS 229 Machine Learning Course

Tyler Neylon

331.2016

These are notes I'm taking as I review material from Andrew Ng's CS 229 course on machine learning. Specifically, I'm watching these videos and looking at the written notes and assignments posted here. These notes are available in two formats: html and pdf.

I'll organize these notes to correspond with the written notes from the class.

1 On lecture notes 1

The notes in this section are based on lecture notes 1.

1.1 Gradient descent in general

Given a cost function $J(\theta)$, the general form of an update is

$$\theta_j := \theta_j - \alpha \frac{\partial J}{\partial \theta_j}.$$

It bothers me that α is an arbitrary parameter. What is the best way to choose this parameter? Intuitively, it could be chosen based on some estimate or actual value of the second derivative of J. What can be theoretically guaranteed about the rate of convergence under appropriate conditions?

Why not use Newton's method? A general guess: the second derivative of J becomes cumbersome to work with.

It seems worthwhile to keep my eye open for opportunities to apply improved optimization algorithms in specific cases.

1.2 Gradient descent on linear regression

I realize this is a toy problem because linear regression in practice is not solve iteratively, but it seems worth understanding well. The general update equation is, for a single example i,

$$\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}.$$

The delta makes sense in that it is proportional to the error $y - h_\theta$, and in that the sign of the product $(y - h_\theta)x$ guarantees moving in the right direction. However, my first guess would be that the expression $(y - h_\theta)/x$ would provide a better update.

For example, suppose we have a single data point (x, y) where $x \neq 0$, and a random value of θ. Then a great update would be

$$\theta_1 := \theta_0 + (y - \theta_0 x)/x,$$

since the next hypothesis value h_θ would then be

$$h_\theta = \theta_1 x = \theta_0 x + y - \theta_0 x = y,$$

which is good. Another intuitive perspective is that we should be making *bigger* changes to θ_j when x_j is *small*, since it's harder to influence h_θ for such x values.

This is not yet a solidified intuition. I'd be interested in revisiting this question if I have time.

1.3 Properties of the trace operator

The trace of a square matrix obeys the nice property that

$$\text{tr } AB = \text{tr } BA. \tag{1}$$

One way to see this is to note that

$$\text{tr } AB = a_{ij}b_{ji} = \text{tr } BA,$$

where I'm using the informal shorthand notation that a variable repeated within a single product implies that the sum is taken over all relevant values of that variable. Specifically,

$$a_{ij}b_{ji} \text{ means } \sum_{i,j} a_{ij}b_{ji}.$$

I wonder if there's a more elegant way to verify (1).

Ng gives other interesting trace-based equations, examined next.

- Goal: $\nabla_A \text{tr } AB = B^T$.

Since

$$\text{tr } AB = a_{ij}b_{ji},$$

we have that

$$(\nabla_A \text{tr } AB)_{ij} = b_{ji},$$

verifying the goal.

- Goal: $\nabla_{A^T} f(A) = (\nabla_A f(A))^T$.

This can be viewed as

$$(\nabla_{A^T} f(A))_{ij} = \frac{\partial f}{\partial a_{ji}} = (\nabla_A f(A))_{ji}.$$

- Goal: $\nabla_A \text{tr}(ABA^TC) = CAB + C^TAB^T$.

I'll use some nonstandard index variable names below because I think it will help avoid possible confusion. Start with

$$(ABA^TC)_{xy} = a_{xz}b_{zw}a_{vw}c_{vy}.$$

Take the trace of that to arrive at

$$\alpha = \text{tr}(ABA^TC) = a_{xz}b_{zw}a_{vw}c_{vx}.$$

Use the product rule to find $\frac{\partial \alpha}{\partial a_{ij}}$. You can think of this as, in the equation above, first setting $xz = ij$ for one part of the product rule output, and then setting $vw = ij$ for the other part. The result is

$$(\nabla_A \alpha)_{ij} = b_{jw}a_{vw}c_{vi} + a_{xz}b_{zj}c_{ix} = c_{vi}a_{vw}b_{jw} + c_{ix}a_{xz}b_{zj}.$$

(The second equality above is based on the fact that we're free to rearrange terms within products in the repeated-index notation being used. Such rearrangement is commutativity of numbers, not of matrices.)

This last expression is exactly the ij^{th} entry of the matrix $CAB + C^T AB^T$, which was the goal.

1.4 Derivation of the normal equation

Ng starts with

$$\nabla_\theta J(\theta) = \nabla_\theta \frac{1}{2}(X\theta - y)^T (X\theta - y),$$

and uses some trace tricks to get to

$$X^T X\theta - X^T y.$$

I thought that the trace tricks were not great in the sense that if I were faced with this problem it would feel like a clever trick out of thin air to use the trace (perhaps due to my own lack of experience with matrix derivatives?), and in the sense that the connection between the definition of $J(\theta)$ and the result is not clear.

Next is another approach.

Start with $\nabla_\theta J(\theta) = \nabla_\theta \frac{1}{2}(\theta^T Z\theta - 2y^T X\theta)$; where $Z = X^T X$, and the doubled term is a valid combination of the two similar terms since they're both real numbers, so we can safely take the transpose of one of them to add them together.

The left term, $\theta^T Z\theta$, can be expressed as $w = \theta_{i1} Z_{ij} \theta_{j1}$, treating θ as an $(n+1)\times 1$ matrix. Then $(\nabla_\theta w)_{i1} = Z_{ij}\theta_{j1} + \theta_{j1} Z_{ji}$, using the product rule; so $\nabla_\theta w = 2Z\theta$, using that Z is symmetric.

The right term, $v = y^T X\theta = y_{i1} X_{ij} \theta_{j1}$ with $(\nabla_\theta v)_{i1} = y_{j1} X_{ji}$ so that $\nabla_\theta v = X^T y$.

These all lead to $\nabla_\theta J(\theta) = X^T X\theta - X^T y$ as before. I think it's clearer, though, once you grok the sense that

$$\begin{aligned} \nabla_\theta \theta^T Z\theta &= Z\theta + (\theta^T Z)^T, \text{and} \\ \nabla_\theta u^T \theta &= u, \end{aligned}$$

both as intuitively straightforward matrix versions of derivative properties.

I suspect this can be made even cleaner since the general product rule

$$\nabla(f \cdot g) = (\nabla f) \cdot g + f \cdot (\nabla g)$$

holds, even in cases where the product fg is not a scalar; e.g., that it is vector- or matrix-valued. But I'm not going to dive into that right now.

Also note that X^TX can easily be singular. A simple example is $X = 0$, the scalar value. However, if X is $m \times n$ with rank n, then $X^TXe_i = X^Tx^{(i)} \neq 0$ since $\langle x^{(i)}, x^{(i)} \rangle \neq 0$. (If $\langle x^{(i)}, x^{(i)} \rangle = 0$ then X could not have rank n.) So X^TX is nonsingular iff X has rank n.

Ng says something in a lecture (either 2 or 3) that implied that $(X^TX)^{-1}X^T$ is *not* the pseudoinverse of X, but for any real-valued full-rank matrix, it *is*.

1.5 A probabilistic motivation for least squares

This motivation for least squares makes sense when the error is given by i.i.d. normal curves, and often this may seem like a natural assumption based on the central limit theorem.

However, this same line of argument could be used to justify any cost function of the form

$$J(\theta) = \sum_i f(\theta, x^{(i)}, y^{(i)}),$$

where f is intuitively some measure of distance between $h_\theta(x^{(i)})$ and $y^{(i)}$. Specifically, model the error term $\varepsilon^{(i)}$ as having the probability density function $e^{-f(\theta, x^{(i)}, y^{(i)})}$. This is intuitively reasonable when f has the aforementioned distance-like property, since error values are likely near zero and unlikely far from zero. Then the log likelihood function is

$$\ell(\theta) = \log L(\theta) = \log \prod_i e^{-f(\theta, x^{(i)}, y^{(i)})} = \sum_i -f(\theta, x^{(i)}, y^{(i)}),$$

so that maximizing $L(\theta)$ is the same as minimizing the $J(\theta)$ defined in terms of this arbitrary function f. To be perfectly clear, the motivation Ng provides only works when you have good reason to believe the error terms are indeed normal. At the same time, using a nice simple algorithm like least squares is quite practical even if you don't have a great model for your error terms.

1.6 Locally weighted linear regression (LWR)

This idea is that, given a value x, we can choose θ to minimize the modified cost function

$$\sum_i w^{(i)}(y^{(i)} - \theta^T x^{(i)})^2,$$

where

$$w^{(i)} = \exp\left(-\frac{(x^{(i)} - x)^2}{2\tau^2}\right).$$

I wonder: Why not just compute the value

$$y = \frac{\sum_i w^{(i)} y^{(i)}}{\sum_i w^{(i)}}$$

instead?

I don't have a strong intuition for which approach would be better, although the linear interpolation is more work (unless I'm missing a clever way to implement LWR that wouldn't also work for the simpler equation immediately above). This also reminds me of the $k-$nearest neighbors algorithm, though I've seen that presented as a classification approach while LWR is regression. Nonetheless, perhaps one could apply a locality-sensitive hash to quickly approximately find the k nearest neighbors and then build a regression model using that.

1.7 Logistic regression

This approach uses a hypothesis function of the form

$$h_\theta(x) = g(\theta^T x),$$

where

$$g(z) = 1/(1 + e^{-z}).$$

The update equation from gradient descent turns out to be nice for this setup. However, besides "niceness," it's not obvious to me why the logistic function g is signifcantly better than any other sigmoid function.

In general, suppose

$$h_\theta(x) = \tau(\theta^T x),$$

where τ is any sigmoid function. Then

$$[\nabla_\theta \ell(\theta)]_j = \sum_i \left(\frac{y^{(i)}}{h^{(i)}} - \frac{1 - y^{(i)}}{1 - h^{(i)}}\right) \frac{\partial \tau^{(i)}}{\partial \theta_j}.$$

Here, $\tau^{(i)}$ indicates the function τ evaluated on θ and $x^{(i)}$. We can split up the term inside the sum like so:

$$a^{(i)} = \frac{y^{(i)}}{h^{(i)}} - \frac{1 - y^{(i)}}{1 - h^{(i)}} = \frac{y^{(i)} - h^{(i)}}{h^{(i)}(1 - h^{(i)})},$$

and

$$b^{(i)} = \frac{\partial \tau^{(i)}}{\partial \theta_j}.$$

Intuitively, the value of $a^{(i)}$ makes sense as it's the error of $h^{(i)}$ weighted more heavily when a wrong output value of h is given with higher confidence. The value $b^{(i)}$ makes sense as the direction in which we'd want to move θ_j in order to increase $\tau^{(i)}$. Multiplied by the sign of $a^{(i)}$, the end result is a movement of θ_j that decreases the error.

It's not obvious to me which choice of function τ is best. Consider, for example, the alternative function

$$\tau(z) = \frac{\arctan(z)}{\pi} + \frac{1}{2}.$$

In this case, when $z = \theta^T x$,

$$b^{(i)} = \frac{\partial \tau^{(i)}}{\partial \theta_j} = \frac{x_j^{(i)}}{1 + (\theta^T x)^2}.$$

In general, sigmoid functions will tend to give a value of $b^{(i)}$ that is small when $|\theta^T x|$ is large, symmetric around $\theta^T x = 0$, and has the same sign as x_j since $\tau - 1/2$ will be an increasing odd function; the derivative of an increasing odd function is even and positive; and this positive even function will be multiplied by x_j to derive the final value of $b^{(i)}$. The values of $b^{(i)}$ will be small for large $|\theta^T x|$ because $\tau(\theta^T x)$ necessarily flattens out for large input values (since it's monotonic and bounded in $[0, 1]$).