batch update protocol

David Yanni

June 2019

1 Problem statement

We are given a model with a known loss function

$$J_1(\vec{q}, \mathbf{X_1}) = \frac{1}{N}\sum_i^N s^i(\vec{q}, \vec{x}^i) = \frac{1}{N}S_1$$

where $s(\cdot)$ is some sort of error term on the i^{th} data point (i.e. i^{th} row), N is the number of rows, and $\vec{q}$ is a vector of weights. $\mathbf{X_1}$ is the matrix of training data that holds as rows each $\vec{x}^i$ (i.e. it's the *design matrix*). Further, we are given the optimal set of weights $\vec{q}^*$ that minimizes this loss function J_1.

In other words, we have a model with optimal weights trained on the data $\mathbf{X_1}$. Now we are given a new, smaller set of data $\mathbf{X_2}$ and we seek a method to find the new optimal weights $\vec{q}^* *$. Obviously we could just make a new training set out of the union $\mathbf{X_1} \bigcup \mathbf{X_2}$, and then train an entirely new model on that larger training set. However, this is slow and repeats a lot of work. Instead we're going to use a variation of "online learning" to bypass all of this wasted work. From here on I'm going to drop the vector and boldface matrix notation to keep things cleaner.

2 Derivation

If there are n rows in the new data X_2 then our new loss function on all of the data is

$$J_2 = \frac{1}{N+n}\sum_i^{N+n} s^i(q, x^i)$$

which we can separate out into two sums

$$\frac{1}{N+n}\left(S_1(q, X_1) + S_2(q, X_2)\right)$$

The goal is to find q^{**} such that $\frac{\partial}{\partial q}J_2 = 0$. Our approach will be to Taylor expand $\frac{\partial}{\partial q}J_2$ about q^* based on the intuition that if $N \gg n$ then q^{**} will be close to q^*. So we let $q^{**} = q * + \delta q$ where δq is small.

$$\frac{\partial}{\partial q} J_2(q^* + \delta q) = \frac{1}{N+n}\left(\frac{\partial}{\partial q} S_1(q^*, X_1) + \frac{\partial^2}{\partial q^2} S_1(q^*, X_1) \cdot \delta q + \right.$$
$$\left. \frac{\partial}{\partial q} S_2(q^*, X_2) + \frac{\partial^2}{\partial q^2} S_2(q^*, X_2) \cdot \delta q + \mathcal{O}(\delta q^2)\right)$$

δq is small so we drop everything of quadratic and higher order and set the loss to 0

$$0 \approx \frac{1}{N+n}\left(\frac{\partial}{\partial q} S_1(q^*, X_1) + \frac{\partial^2}{\partial q^2} S_1(q^*, X_1) \cdot \delta q + \frac{\partial}{\partial q} S_2(q^*, X_2) + \frac{\partial^2}{\partial q^2} S_2(q^*, X_2) \cdot \delta q\right)$$

Next note that $\frac{\partial}{\partial q} S_1(q^*, X_1) = 0$ (since q^* minimizes J_1). Also $N + n > 0$ so we can lose that prefactor:

$$0 \approx \left(\frac{\partial}{\partial q} S_2(q^*, X_2) + \frac{\partial^2}{\partial q^2}\left(S_1(q^*, X_1) + S_2(q^*, X_2)\right) \cdot \delta q\right)$$

Now since q is a vector $\frac{\partial^2}{\partial q^2} S$ will be a matrix (the Hessian). So let's name the term $\mathbf{M} = \frac{\partial^2}{\partial q^2}\left(S_1(q^*, X_1) + S_2(q^*, X_2)\right)$, and go back to vector/matrix notation for the final step

$$\boxed{\delta\vec{q} = -\mathbf{M}^{-1} \cdot \nabla_q S_2(\vec{q^*}, \mathbf{X_2})}$$

3 Verification

see python file on my GitHub.

4 Comments and references

.

1) This is basically a variant of Newton's method

2) This broadly falls into the category of "Online Learning" if you want to learn more. Link.