Newton's Method and Meta-Fits

G. Mirek Brandt

September 12, 2018

1 Introduction

I start by deriving Newton's method for maximizing a scalar function which depends on one parameter. I then state the conceptually similar analogue for a maximizing a function f which depends continuously on a finite set of parameters $c = c_0, c_1, ...$ Newton's method depends on evaluating the gradient of f and the set of mixed partials with respect to the c_i, contained in the Hessian matrix. If the c_i themselves are described by a meta-function, one can just as easily maximize f by optimizing the coefficients for the polynomial expansion of the meta-function (hereafter called a meta-fitting), and the corresponding Hessian and gradient.

2 Newton's Method

Here I derive Newton's method and the higher dimensional analogues, and apply it to the problem of fitting bright lines, called orders or traces, across a CCD image from an Echelle Spectrograph.

2.1 The 1-Dimensional Case

For a scalar function $f(x)$ and some initial guess at the n^{th} iteration, x_n, we seek the perturbation Δx such that we arrive at a stationary point of the function $f(x)$. We have that

$$f(x_n + \Delta x) = f(x_n) + \Delta x f'(x_n) + \frac{1}{2}\Delta x^2 f''(x_n) \tag{1}$$

We require that

$$\frac{d}{d\Delta x} f(x_n + \Delta x) = 0 \tag{2}$$

Thus we have

$$f'(x_n) + \Delta x f''(x_n) = 0 \tag{3}$$

So we seek the Δx which solves $f''(x_n)\Delta x = (-1)f'(x_n)$. Afterwhich, $x_{n+1} = x_n + \Delta x$. This is called Newton's method. One repeats this until satisfactory convergence. Typically this converges quadratically given a close enough initial guess. One can also apply a relaxed newton's method by including a multiplicative factor $\gamma \in (0, 1]$ whereby $x_{n+1} = x_n + \gamma \cdot \Delta x$ Note $\gamma = 1$ is standard Newton's method.

2.2 N-Dimensions

In N-dimensions, we seek the perturbation to the N parameters $c = c_0, c_1, .., c_{N-1}$. We thus extend $f'(c) \to \vec{\nabla} f(c)$, the gradient, and $f''(c) \to H[f(c)]$, the Hessian. For N parameters, the Hessian matrix has dimension $N \times N$ with each element

$$H_{j,k} = \frac{\partial^2 f}{\partial c_j \partial c_k} \tag{4}$$

Δx becomes N dimensional as well.

3 Application to Trace-Fitting of Spectroscopic Images

We would like to fit the smooth lines of brightness, each one called an order or trace, which show up on CCD images collected by an Echelle-Spectrograph. At a given x-pixel coordinate, the only degree of freedom is the y-pixel coordinate. Let $y_i = T_d(i)$ where i is the x-pixel coordinate and y_i is the y value of our guess for the d^{th} order location evaluated at i, described by the function $T_d(i)$. There are typically 70 or so orders visible per image. At each pixel, we seek to maximize the flux $f(y_i)$ via Newton's method. Assume

$$y_i = T_d(y_i) = c_0 + c_1 P_1(i) + ... \tag{5}$$

where $P_j(i)$ is the j^th order Legendre polynomial evaluated at i. I.e. the trace is decomposable as a sum of Legendre polynomials. By the chain rule we have that

$$H(y_i)_{j,k} = \frac{\partial^2 f}{\partial y_i^2}\frac{\partial y_i}{\partial c_j}\frac{\partial y_i}{\partial c_k} \tag{6}$$

Because the flux is positive definite (ignoring noise), the Hessian for the entire trace is simply the sum of the Hessians for each x-pixel across the trace. Thus

$$H_{j,k} = \sum_i \frac{\partial^2 f}{\partial y_i^2}\frac{\partial y_i}{\partial c_j}\frac{\partial y_i}{\partial c_k} \tag{7}$$

To be austere, we include the index of the trace order d, as each trace has a different set of coefficients c_k which describe its path across the CCD.

$$H(d)_{j,k} = \sum_i \frac{\partial^2 f}{\partial y_i^2}\frac{\partial y_i}{\partial c_{j,d}}\frac{\partial y_i}{\partial c_{k,d}} \tag{8}$$

Where $y_i = T_d(y_i) = c_{0,d} + c_{1,d}P_1(i) + ...$ Moreover from inspection of equation 5, one sees that $\partial y_i/\partial c_{j,d} = P_j(i)\ \forall\ d$. The components of the gradient are simply

$$\vec{\nabla} f_{k,d} = \sum_i \frac{\partial f}{\partial y_i}\frac{\partial y_i}{\partial c_{k,d}} = \sum_i \frac{\partial f}{\partial y_i}P_k(i) \tag{9}$$

Thus if we were independently solving for the optimum coefficients per-trace, we would solve for the pertubation vector $\vec{\Delta c}$ such that

$$H\vec{\Delta c} = (-1)\cdot\vec{\nabla} f \tag{10}$$

Where fixed d is assumed. In python, one would iteratively perform

```
import numpy as np
#  Evaluate H and the gradient using the initial coeffs guess. Then:
deltacoeffs = np.linalg.solve(H,(-1)*grad)
coeffs += deltacoeffs
```

One can get $\partial f/\partial y_i$ and $\partial^2 f/\partial y_i^2$ from spline interpolations of the CCD image.

3.1 The Meta-Fit

It turns out that each of the coefficients of the orders (for a single fiber) can be described by smooth polynomials as a function of order number. Specifically for a meta-fit of order M, we have

$$c_{k,d} = \alpha_{0,k} + \alpha_{1,k}\cdot d + \alpha_{2,k}\cdot d^2 + ... \tag{11}$$

One could use Legendre polynomials as the basis [1] like we did with the traces, or use standard polynomials. I chose to use Legendre polynomials as the basis in my implementation, however in this document I use powers of d, i.e. ordinary polynomials. In order to optimize the $\alpha_{p,k}$ [2] to

[1] Then $d^j \rightarrow P_j(d)$

[2] $k \leq N+1$ and $p \leq M+1$

maximize the flux across all traces, we use the chain rule again. We start with the gradient for the d^{th} trace, which has elements

$$\frac{\partial f(d)}{\partial \alpha_{p,k}} = \sum_i \frac{\partial f}{\partial y_i}\frac{\partial y_i}{\partial c_{k,d}}\frac{\partial c_{k,d}}{\partial \alpha_{p,k}} = \sum_i \frac{\partial f}{\partial y_i} P_k(i) d^p \tag{12}$$

Note one can enumerate the alpha coefficients in a single vector via $\vec{\alpha} = [\alpha_{0,0}, \alpha_{1,0}, ..., \alpha_{M,N-1}, \alpha_{M,N}]$. In fact, we use this arbitrary ordering to construct both the gradient and the Hessian. The meta-gradient is arranged as

$$\mathscr{M}\nabla_d f = (\frac{\partial f(d)}{\partial \alpha_{0,0}}, \frac{\partial f(d)}{\partial \alpha_{1,0}}, \frac{\partial f(d)}{\partial \alpha_{0,1}}, \frac{\partial f(d)}{\partial \alpha_{1,1}} ...) \tag{13}$$

As discussed before. The rows and columns of the Hessian we will build have the same order as well.

For a single trace, the meta-fit Hessian elements follow from the chain rule.

$$\mathscr{H}(d)_{p,j,q,k} = \frac{\partial^2 f(d)}{\partial \alpha_{p,j}\partial \alpha_{q,k}} = \sum_i \frac{\partial^2 f}{\partial y_i^2}\frac{\partial y_i}{\partial c_{j,d}}\frac{\partial y_i}{\partial c_{k,d}}\frac{\partial c_{j,d}}{\partial \alpha_{p,j}}\frac{\partial c_{k,d}}{\partial \alpha_{q,k}} \tag{14}$$

$$= H(d)_{j,k}\frac{\partial c_{j,d}}{\partial \alpha_{p,j}}\frac{\partial c_{k,d}}{\partial \alpha_{q,k}} \tag{15}$$

$$= H(d)_{j,k} d^p d^q \tag{16}$$

And the elements of the meta Hessian are re-arranged [3] such that

$$\mathscr{H}(d) = \begin{pmatrix} \mathscr{H}(d)_{0,0,0,0} & \mathscr{H}(d)_{0,0,1,0} & \cdots \\ \mathscr{H}(d)_{1,0,0,0} & \mathscr{H}(d)_{1,0,1,0} & \\ \vdots & & \ddots \end{pmatrix} \tag{17}$$

Note the ordering of the parameters in the Hessian and gradient is arbitrary, it is just imperative that they agree with each other. To construct the full meta Hessian and gradient, we sum over all the orders. Thus

$$\mathscr{H} = \sum_d \mathscr{H}(d) \tag{18}$$

$$\mathscr{M}\nabla f = \sum_d \mathscr{M}\nabla_d f \tag{19}$$

Now we solve

$$\mathscr{H} = (-1)\cdot \mathscr{M}\nabla f \vec{\Delta\alpha} \tag{20}$$

And add either the strict perturbation $\vec{\Delta\alpha}$ or the relaxed $\gamma \cdot \vec{\Delta\alpha}$ to $\vec{\alpha}_n$ to obtain $\vec{\alpha}_{n+1}$ for the next iterations guess. In practice, it is more numerical stable to feed the hessian and the gradient into scipy.optimize NEWTON-CG.

[3] To implement this in python, I first constructed the Hessian as a $N \cdot M$ length list, then reshaped it into the appropriate matrix.