Enzyme Kinetics

An application of gradient descent and Newton's method to solving problems of parameter estimation

BENG 221-PROBLEM SOLVING SESSION

11/12/2010

INTRODUCTION

With advances in understanding in a variety of scientific disciplines, as occur with time, our ability to reduce to mathematical terms the world in which we live poses new challenges. Whereas relatively elementary mathematical methods could be applied to fitting models of simple systems, the task of parameter estimation becomes more complex as we attempt to construct higher-order system models. Nowhere is this as apparent as in the realm of biology, in which biologic systems are comprised of intricately woven feed-forward and feedback electrochemical networks. Although one can approach such problems in a reductionist manner, over-simplification could potentially translate into the loss of physiologic significance. Therefore, as Bioengineering seeks to contribute ultimately towards clinical understanding, bioengineers must ultimately turn towards methods capable of tackling these more advanced models. Both gradient descent and Newton's method can be applied to just such an end, in which what one seeks to minimize is an optimization function associated with the model of interest. As an example, these techniques are applied herein to solving the issue of model fitting in enzyme kinetics.

$$E + S \underset{k_{-1}}{\overset{k_1}{\rightleftarrows}} ES \xrightarrow{k_2} E + P$$

Figure 1. Reaction pathway for a single enzyme/single substrate reaction. E = Enzyme, S = substrate, and E-S represents the enzyme substrate complex

The basics of enzyme kinetic theory are best understood through the work of Leonor Michaelis and Maud Menten. In the case of single-substrate enzymatic reactions, the chemical formulation is shown in Figure 1, where K_1, K_{-1}, and K_2 are the reaction rate constants. The expression for the rate of formation of product is then given by Eq. 1. Likewise, the rate of formation of the enzyme-substrate complex is given by Eq. 2, which is assumed to be zero. In other words, over time the enzyme-substrate complex is degraded at a rate equal to its formation. Lastly, we can define the total amount of enzyme in the system through Eq. 3, which assumed to remain constant. Now with three equations and three unknowns (the undetermined rate constants), the system can be solved such that Eq. 4 is the result. Eq. 4 is known as the Michaelis-Menten equation, and gives the initial rate of product formation as a function of the maximum enzymatic rate, the substrate concentration and a rate constant Km (which itself is a function of the elementary rate constants). Km reflects the enzyme affinity for the substrate.

$$\frac{dP}{dt} = k_2[ES] \quad (1)$$

$$\frac{dES}{dt} = k_1[E][S] - (k_2 + k_{-1})[ES] \stackrel{\text{def}}{=} 0 \quad (2)$$

$$E_0 = ES + E \stackrel{\text{def}}{=} constant \quad (3)$$

$$v_0 = \frac{v_{max}[S]}{K_M + [S]} \quad , \quad K_M = \frac{k_{-1} + k_2}{k_1} \quad (4)$$

The above formulation is valid for simple enzyme/substrate interactions, however often a point of interest is the effect of an inhibitor on the enzyme affinity. It can be shown in a similar manner that the effect of a reversible competitive inhibitor is to modify the apparent Km of the enzyme/substrate interaction by a term α, where [I] is the concentration of inhibitor, and KI is a term describing the affinity of the enzyme for the inhibitor. By conducting enzyme kinetic assays, in which the initial rate of product formation v_0 is measured against varying concentrations of substrate, data for the model can be generated so as to provide a means to fit the parameters Km and Vmax.

$$v_0 = \frac{v_{max}[S]}{\alpha K_M + [S]} \quad , \quad \alpha = 1 + \frac{[I]}{KI} \qquad (5)$$

Theory

The method of gradient descent and Newton's method are both numerical methods of finding local maxima and minima of multivariable equations. Technically either may be applied to any continuous and differentiable function of an arbitrary number of variables and they differ principally in the rate at which they converge to local extrema.

The method of gradient descent is a first order approach to finding local extrema and will hence be treated here first. The gradient $\boldsymbol{f} = \boldsymbol{\nabla} F(x_1, x_2, \dots, x_n)$ is defined as a vector $\boldsymbol{f} = (f_1, f_2, \dots f_n)$ for which each component $f_i = \frac{\partial F}{\partial x_i}$. The property that makes the gradient useful to optimization problems is that the gradient vector points in the direction of maximum increase of the function $F(x_1, x_2, \dots x_n)$. This may be demonstrated by observing that $dF = \frac{\partial F}{\partial x_1} dx_1 + \frac{\partial F}{\partial x_2} dx_2 + \cdots + \frac{\partial F}{\partial x_n} dx_n = \boldsymbol{\nabla} F(x_1, x_2, \dots, x_n) \cdot \boldsymbol{dr}$ where $\boldsymbol{dr} = (dx_1, dx_2, dx_3)$ represents a small deviation in an arbitrary direction away from $(x_1, x_2, \dots x_n)$. As $\boldsymbol{\nabla} F(x_1, x_2, \dots, x_n) \cdot \boldsymbol{dr} = dF = |\boldsymbol{\nabla}\mathrm{F}||\mathbf{dr}|\cos(\theta)$ where θ is the angle between the two vectors, one can see that dF is maximized for $\theta = 0$ and minimized for $\theta = \pi$. Thus the change in F is maximized by moving along the direction of the gradient and minimized by moving in the direction opposite to the gradient.

To apply the gradient descent method to a minimization problem, one may start by defining an initial guess $F(\boldsymbol{x_0})$. If the function $F(\boldsymbol{x})$ is differentiable and $\gamma > 0$ is small, then $F\big(\boldsymbol{x_0} - \gamma \boldsymbol{f}(\boldsymbol{x_0})\big) = F(\boldsymbol{x_1}) \leq F(\boldsymbol{x_0})$. To mimimize a function $F(\boldsymbol{x})$ by gradient descent one must repeat the formula

$$\boldsymbol{x_{i+1}} = \boldsymbol{x_i} - \gamma \boldsymbol{f}(\boldsymbol{x_i}) \qquad (6)$$

until the minimum of the function is reached. At the point $\boldsymbol{x}^*$ for which $F(\boldsymbol{x})$ is minimized, $\boldsymbol{f}(\boldsymbol{x}^*) = 0$ and the descent will progress no further. The limitations of this method generally concern the amount of time required for the algorithm to reach the minimum. If γ is defined as too small, then it will take a great number of steps to reach the minimum, especially if $\boldsymbol{x_0}$ is far from $\boldsymbol{x}^*$. Conversely if γ is too large, the descent may overstep the minimum, also increasing the convergence time. An additional limitation of this method is that it will only find local minima. Thus $\boldsymbol{x_0}$must be chosen carefully to avoid finding local, but undesired minima.

Newton's method is a second order approach to finding the local extrema of a multi-variable function. It is based on finding the minimum of a locally defined quadratic Taylor series expansion of the function $F(x)$. In one dimension one can approximate $F(x_0 + \Delta x) \approx F(x_0) + F'(x_0)\Delta x + \frac{1}{2}F''(x_0)\Delta x^2$. The minimum of this local quadratic function then occurs if $F'(x + \Delta x) \approx F'(x_0) + F''(x_0)\Delta x = 0$, or in other words, if $\Delta x = -\frac{F'(x_0)}{F''(x_0)}$. If the equation is itself quadratic then moving from $x_0 \rightarrow x_1 = x_0 - \frac{F'(x_0)}{F''(x_0)}$ will solve for the minimum in one step. If the equation is merely twice differentiable, then repeating this algorithm iteratively will eventually lead to the convergence of $x_0 \rightarrow x^*$, the local minimum of $F(x)$. Expanding this algorithm to functions of multiple variables is straightforward. In the multivariable case

$$F(\boldsymbol{x_0} + \Delta\boldsymbol{x}) \approx F(\boldsymbol{x_0}) + \boldsymbol{\nabla} F(\boldsymbol{x_0})\boldsymbol{\Delta x} + \frac{1}{2}\boldsymbol{\Delta x^T H}(\boldsymbol{x_0})\boldsymbol{\Delta x},$$

$$\text{where } \boldsymbol{H}(\boldsymbol{x_0}) = \begin{bmatrix} \frac{\boldsymbol{\delta^2 F}}{\boldsymbol{\delta {x_1}^2}} & \cdots & \frac{\boldsymbol{\delta^2 F}}{\boldsymbol{\delta x_1 \delta x_n}} \\ \vdots & \ddots & \vdots \\ \frac{\boldsymbol{\delta^2 F}}{\boldsymbol{\delta x_1 \delta x_n}} & \cdots & \frac{\boldsymbol{\delta^2 F}}{\boldsymbol{\delta {x_n}^2}} \end{bmatrix}$$

is the second order equivalent of the gradient known as the Hessian. The equivalent update rule for this case is

$$\boldsymbol{x_{i+1}} = \boldsymbol{x_i} - [\boldsymbol{H}(\boldsymbol{x_i})]^{-1}\boldsymbol{\nabla} F(\boldsymbol{x_i}) \qquad (7)$$

Note that this update rule will lead to the local minimum only if $[\boldsymbol{H}(\boldsymbol{x_i})]$ is positive definite, meaning that $\boldsymbol{\Delta x^T H}(\boldsymbol{x_i})\boldsymbol{\Delta x} > 0$ for all real, non-zero $\boldsymbol{\Delta x}$. This is equivalent to stating that the quadratic function described by the second-order Taylor expansion must be convex and have a local minimum. If $[\boldsymbol{H}(\boldsymbol{x_i})]$ is negative definite, the update rule will lead to a local maximum.

Application to Michaelis-Menten kinetics

As discussed above, the Michaelis-Menten equation

$$V = V_{max}\frac{[S]}{K_m + [S]}$$

describes the rate of an enzymatic reaction as a function of the concentration of its substrate $[S]$. There are two additional parameters $K_m$and V_{max}which are intrinsic to the enzyme-substrate reaction and represent the binding affinity of the enzyme for the substrate and the maximum reaction rate, respectively. Bio-chemists characterize enzymes of interest by these properties and do so by measuring reaction rate with increasing concentrations of substrate. They then attempt to fit their data to the general equation by choosing values of $K_m$and V_{max} that minimize the difference between their results and those predicted by the model. This is known as least squares analysis. In this case we have two sets of data. The vector $\boldsymbol{s}$ represents measured concentrations of substrate and the vector $\boldsymbol{h}$ represents measured values of reaction velocity. The vector $\mathbf{v} = V_{max}\frac{\boldsymbol{s}}{K_m+\boldsymbol{s}}$ represents the predicted velocity given the model. We can then define a cost function

$$C(\boldsymbol{h}, \boldsymbol{v}(\boldsymbol{s})) = (\boldsymbol{v}-\boldsymbol{h})^2 = (\boldsymbol{v}-\boldsymbol{h})\cdot(\boldsymbol{v}-\boldsymbol{h}) = \sum_{\boldsymbol{i}}^{\boldsymbol{n}}\left(V_{max}\frac{s_i}{K_m+s_i} - h_i\right)^{\mathbf{2}} \qquad (8)$$

In theory one can minimize this particular equation completely analytically by converting the Michaelis-Menten equation into the Lineweaver-Burke equation $\frac{1}{V} = \frac{K_m}{V_{max}}\cdot\frac{1}{[S]} + \frac{1}{V_{max}}$. This version of the equation is linear, and $K_m$and V_{max} can found using a general linear model approach. In this case we would write

$$\boldsymbol{y} = c_1\boldsymbol{x} + c_2\boldsymbol{b} + \boldsymbol{n} \qquad (9)$$

where $c_1 = \frac{K_m}{V_{max}}, \boldsymbol{x} = \frac{1}{\boldsymbol{s}}$, $\boldsymbol{y} = \frac{1}{\boldsymbol{h}}, c_2 = \frac{1}{V_{max}}$, **b** is a constant vector, and **n** is random noise in the measurement. For zero **n**, $[\boldsymbol{x}\ \boldsymbol{b}]\begin{bmatrix}c_1\\ \boldsymbol{c_2}\end{bmatrix} = \boldsymbol{y}$, but for non zero **n**, $[\boldsymbol{x}\ \boldsymbol{b}]\begin{bmatrix}c_1\\ \boldsymbol{c_2}\end{bmatrix} \neq \boldsymbol{y}$. However, the error $([\boldsymbol{x}\ \boldsymbol{b}]\begin{bmatrix}c_1\\ c_2\end{bmatrix} - \boldsymbol{y})^{\mathbf{2}}$ can be minimized if $[\boldsymbol{x}\ \boldsymbol{b}]\begin{bmatrix}c_1\\ c_2\end{bmatrix} - \boldsymbol{y} = \mathbf{n}$, that is if $\begin{bmatrix}c_1\\ c_2\end{bmatrix}$ are chosen such that only random

noise separates the data from the model. If **n** is gaussian, and then$[\boldsymbol{x}\ \boldsymbol{b}]^T\boldsymbol{n} = 0$, so $[\boldsymbol{x}\ \boldsymbol{b}]^T\left([\boldsymbol{x}\ \boldsymbol{b}]\begin{bmatrix}c_1\\c_2\end{bmatrix} - \boldsymbol{y}\right) = \boldsymbol{0}$. This means that $[\boldsymbol{x}\ \boldsymbol{b}]^T[\boldsymbol{x}\ \boldsymbol{b}]\begin{bmatrix}c_1\\c_2\end{bmatrix} = [\boldsymbol{x}\ \boldsymbol{b}]^T\boldsymbol{y}$ and

$$([\boldsymbol{x}\ \boldsymbol{b}]^T[\boldsymbol{x}\ \boldsymbol{b}])^{-\boldsymbol{1}}[\boldsymbol{x}\ \boldsymbol{b}]^T[\boldsymbol{x}\ \boldsymbol{b}]\begin{bmatrix}c_1\\c_2\end{bmatrix} = ([\boldsymbol{x}\ \boldsymbol{b}]^T[\boldsymbol{x}\ \boldsymbol{b}])^{-\boldsymbol{1}}[\boldsymbol{x}\ \boldsymbol{b}]^T\boldsymbol{y} = \begin{bmatrix}c_1\\c_2\end{bmatrix} \qquad (10)$$

which solves for the optimum values of $c_1 = \frac{K_m}{V_{max}}\ and\ c_2 = \frac{1}{V_{max}}$. We can demonstrate this for real set of data taken from Schropp *et al*. In this reaction we have an enzyme called *invertase* converting sucrose to glucose and fructose. The substrate concentration and measured reaction velocities are given by

$$\mathbf{s}^{\mathbf{T}} = [0.3330\ 0.1670\ 0.0833\ 0.0416\ 0.0208\ 0.0104\ 0.0052]$$

$$\mathbf{h}^{\mathbf{T}} = [3.636\ 3.636\ 3.236\ 2.666\ 2.114\ 1.466\ 0.866]$$

For this reaction the true $K_m = 3.91mM$ and $V_{max} = 0.0179s^{-1}$. Using the Lineweaver-Burke equation and the general linear model with

$$\mathbf{x}^{\mathbf{T}} = [3.0030\quad 5.9880\quad 12.0048\quad 24.0385\quad 48.0769\quad 96.1538\quad 192.3077]$$

$$\mathbf{y}^{\mathbf{T}} = [0.2750\quad 0.2750\quad 0.3090\quad 0.3751\quad 0.4730\quad 0.6821\quad 1.1547]$$

$$\mathbf{b}^{\mathbf{T}} = [1\quad 1\quad 1\quad 1\quad 1\quad 1\quad 1]$$

Plugging these values into the equation for $\begin{bmatrix}c_1\\c_2\end{bmatrix}$ we get $\begin{bmatrix}c_1\\c_2\end{bmatrix} = \begin{bmatrix}0.0046\\0.2531\end{bmatrix}$ or $K_m = 3.95$ and $V_{max} = 0.0182$. Unfortunately, because linearizing the Michaelis-Menten equation requires inversion of the values for both the independent variable **s** and the dependent variable **y**, this method is prone to produce false results if the errors in the measurements are very large, and it is thus rarely used for parameter estimation.

The gradient descent and Newton's methods are more robust methods of error estimation in this case. Returning to the cost function

$$C(\boldsymbol{h}, \boldsymbol{v}(\boldsymbol{s})) = \sum_{\boldsymbol{i}}^{\boldsymbol{n}}\left(V_{max}\frac{s_i}{K_m + s_i} - h_i\right)^{\boldsymbol{2}},$$

we need to find $\nabla C(\boldsymbol{h}, \boldsymbol{v}(\boldsymbol{s}))$ and $\boldsymbol{H}[C(\boldsymbol{h}, \boldsymbol{v}(\boldsymbol{s}))]$.

$$\nabla C = (\sum_{\boldsymbol{i}}^{\boldsymbol{n}} 2\left(V_{max}\frac{s_i}{K_m+s_i} - h_i\right)\cdot\frac{s_i}{K_m+s_i},\ \sum_{\boldsymbol{i}}^{\boldsymbol{n}} -2\left(V_{max}\frac{s_i}{K_m+s_i} - h_i\right)\cdot V_{max}\frac{s_i}{(K_m+s_i)^2})$$

and

$$\boldsymbol{H}[C] = \begin{bmatrix} \sum 2(\frac{s_i}{K_m+s_i})^2 & \sum -2V_{max}\frac{{s_i}^2}{(K_m+s_i)^3} - \sum_{\boldsymbol{i}}^{\boldsymbol{n}} 2\left(V_{max}\frac{s_i}{K_m+s_i} - h_i\right)\frac{s_i}{(K_m+s_i)^2} \\ \sum -2V_{max}\frac{{s_i}^2}{(K_m+s_i)^3} - \sum_{\boldsymbol{i}}^{\boldsymbol{n}} 2\left(V_{max}\frac{s_i}{K_m+s_i} - h_i\right)\frac{s_i}{(K_m+s_i)^2} & \sum 2\left({V_{max}}^2\frac{{s_i}^2}{(K_m+s_i)^4}\right) + 4V_{max}\left(V_{max}\frac{s_i}{K_m+s_i} - h_i\right)\frac{s_i}{(K_m+s_i)^3} \end{bmatrix}$$

This is the limit of what can be solved analytically for these methods. From this point on we will have to choose a "best guess" of what we believe $(w_0^1, w_0^2) = (K_m, V_{\max})$ to be and apply the update rule

$$(w_1^1, w_1^2) = (w_0^1, w_0^2) - \gamma \boldsymbol{\nabla} C(w_0^1, w_0^2) \qquad (11)$$

for gradient descent and

$$(w_1^1, w_1^2) = (w_0^1, w_0^2) - \left[\boldsymbol{H}\left(C(w_0^1, w_0^2)\right)\right]^{-\boldsymbol{1}} \boldsymbol{\nabla} C(w_0^1, w_0^2) \qquad (12)$$

for Newton's Method, until the local minimum is reached.

Application to reactions with competitive inhibitors

As discussed above, in the presence of a competitive inhibitor, the Michaelis-Menten equation becomes

$$V = V_{max} \frac{[S]}{(1+\frac{[I]}{K_I})\, K_m + [S]}$$

where [I] is the concentration of the inhibitor, and K_I describes the affinity of the inhibitor for the enzyme. Now to find the best estimate of the parameters we need to do a least squares fit with three variables. Already this equation cannot be linearized like the original equation, so we will have to use a gradient descent or Newton's method optimization algorithm. Our new cost function is

$$C(\boldsymbol{h}, \boldsymbol{v}(\boldsymbol{s}, \boldsymbol{i})) = \sum_{\boldsymbol{i}}^{\boldsymbol{n}} \left(V_{max} \frac{s_i}{(1+\frac{i_i}{K_I})\, K_m + s_i} - h_i \right)^{\mathbf{2}} \qquad (13)$$

and its gradient is

$$\nabla C(K_m, K_I, V_{max}) = \left(\sum 2 \left(V_{max} \frac{s_i}{\left(1+\frac{i_i}{K_I}\right) K_m + s_i} - h_i \right) V_{max} \frac{s_i}{\left(\left(1+\frac{i_i}{K_I}\right) K_m + s_i\right)^2} \left(1+\frac{i_i}{K_I}\right), \right.$$
$$\sum 2 \left(V_{max} \frac{s_i}{\left(1+\frac{i_i}{K_I}\right) K_m + s_i} - h_i \right) V_{max} \frac{s_i}{\left(\left(1+\frac{i_i}{K_I}\right) K_m + s_i\right)^2} \left(\frac{i_i}{{K_I}^2}\right) K_m,$$
$$\left. \sum 2 \left(V_{max} \frac{s_i}{\left(1+\frac{i_i}{K_I}\right) K_m + s_i} - h_i \right) \left(\frac{s_i}{\left(1+\frac{i_i}{K_I}\right) K_m + s_i} \right) \right)$$

Already, with only three parameters, the Hessian becomes quite impractical to compute analytically. Fortunately, there are automated computational techniques for analytically determining the gradient and Hessian that allow the gradient descent and Newton's method to be employed to higher order problems.

Conclusion

For this project we attempted to use several numerical methods as a means of doing parameter estimation. As a particular example, we chose to use a Michaelis-Menten type problem and use data from literature to do parameter estimation. The parameters that were estimated were K_{max} and V_m. The linear analytical solution was first found and was used as a means to compare the numerical answers to. Both Newton's method and a gradient descent method were used to estimate the parameters, and both produced values that agreed with the analytical solution. This allowed us to conclude that these methods are valid means of doing parameter estimation, at least in this particular case. Since we determined that both methods were valid for parameter estimation, the next step was to compare their efficiencies and see which method is more desirable for our example.

Both Newton's method and the gradient descent method had positive and negative aspects to them. The gradient descent method worked well if the initial guess was relatively close to the solution. If the approach to the solution is fairly flat, convergence can take a long time, especially if there is no optimization where a variable time step is used. If the approach to the solution is very sharp, there is a chance that it will be stepped over if using the gradient descent method, which is obviously a problem because the true solution can be missed. With Newton's method, convergence can be very fast if the initial guess is a good one. Inflection points can cause problems because when the Hessian becomes non-invertible, Newton's method will cause the problem to diverge and not reach a solution. A time limiting factor with Newton's method is the fact that the derivative has to be calculated at each time step, which in some cases can be time consuming. Lastly, the initial guess must be sufficiently near the desired root or it could fail to converge; thus the need for multiple initial guesses.

Future work

One of the most substantial improvements that can be made to this project would be to add the capability of using a cluster and parallel computing. This would allow for many more initial guesses without the worry of how much time it will take for the simulation to run. Parallel computing would not only shorten the amount of time needed to run a simulation, but would help insure that the true solution is reached. Many initial guesses increase the chances of finding the global minimum instead of just local a minimum, thus ensuring the desired solution is reached.

Another path that can be taken in the future would be to implement a two-point secant method. This might be particularly useful if more powerful computing is not an option. The secant method is essentially a finite difference approximation of Newton's method. One of its benefits is that it does not have to actually compute the derivative at each time step like Newton's method does, but rather uses a numerical approximation of it. This approximation actually makes the method faster between each time point because it does not have to evaluate the function and its derivative at each step. If parallel computing is used, Newton's method is generally faster because in that situation the function and the derivative can be evaluated at the same time.