INFR10086 Machine Learning (MLG) Semester 2, 2023/4

Tutorial 2: Curve Fitting

In this tutorial, we will write a program that can fit curves, one of the simplest machine learning algorithm.

1 Finding the Dissociation Constant

One of your biochemistry friend approaches you with the data in Table 1. We are looking for a way to find the dissociation constant K_D for hemoglobin binding to oxygen. We know that the disassociation constant K_D appears in the equation[1]

$$\theta = \frac{[L]^n}{K_D + [L]^n}, \tag{1}$$

where θ is the amount of saturation and $[L]$ is the concentration. We know the constant n is 2.49.[2] We also know that K_D should be positive, and want to find K_D given the data in Table 1.[3]

Table 1: The percentage of hemoglobin bound to oxygen (or the amount of saturation) under different oxygen concentration (measured in kPa).

O_2 concentration	saturation
1.333	0.425
1.9995	0.63
2.666	0.755
3.3325	0.844
3.999	0.91
4.6655	0.94
6.665	0.96
13.33	0.99

[1]This equation is called the Hill equation.

[2]The constant n is also known as Hill constant.

[3]These entries are actually taken from Table III in Hill's 1910 paper, "The possible effects of the aggregation of the molecules of hemoglobin on its dissociation curves," J. Physiol. 40 (Suppl).

2 Abstraction

Without knowing all the details of how oxygen binds to hemoglobin, we can abstract away all the chemistry and focus on the problem itself. Equation (1) can be written as

$$y = \frac{x^n}{K + x^n}, \tag{2}$$

where x is the O_2 concentration and y is the saturation. We want to find K such that Equation (1) gives the best fit for Table 1, so this is a curve-fitting problem.

As for many curve-fitting problems, mean-squared error should be a good loss for finding the best fit. The loss for this particular curve is

$$L = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{x_i^n}{K + x_i^n} - y_i \right)^2 . \tag{3}$$

Implementation. Write a script 1) to plot the points in Table 1 and 2) to overlay a curve when $K = 2$ or $K = 3$ on top of the points.

Implementation. Write a script to print out the loss value given a particular K, for example, when $K = 2$ or $K = 3$.

Discussion. Do you think the curves fit better when they have lower mean-squared errors?

Discussion. In what range do you think the optimal K lies?

3 Binary Search

Since this is a one-dimensional problem, we have the luxury to manually try a few values of K and choose the best one. We obviously do not want to do random search. A simple strategy is to do binary search. Similar to searching a number in a sorted array, we split a segment in the middle, and choose the one that might contain the optimal value. To choose whether we want to move right (the larger side) or left (the smaller side), we can look at the first derivative. If the first derivative is negative, that means the optimal value is on the right; if the first derivative is positive, that means the optimal value is on the left.

Discussion. Why do we want to move left (to the smaller side) when we see a positve first derivative?

Table 2: A sequence of values tried when performing a binary search between 0 and 10.

K	$\frac{\partial L}{\partial K}$	L
0	-0.0951	0.0707
10	0.00601	0.032
5	0.00406	0.0047
2.5	-0.00355	0.00145
3.75	0.00177	0.000835
3.125	-0.000296	0.00033

Implementation. The first derivative of the loss function is

$$\frac{\partial L}{\partial K} = \frac{1}{N}\sum_{i=1}^{N} 2\left(\frac{x_i^n}{K + x_i^n} - y_i\right)\left(\frac{x_i^n}{K + x_i^n}\frac{-1}{K + x_i^n}\right). \tag{4}$$

Write a script to compute the derivative of L.

Discussion. Manually do the binary search with the script you write starting from $K = 0$. Can you narrow down the range of optimal K?

Iterating this process a few times is enough to narrow down the optimal K. For example, if we start from the range $[0, 10]$, we end up with the sequence in Table 2. Based on the search, we can guess that the optimal value is between 3.125 and 3.75.

4 Optimization

Another option is to do gradient descent on the loss function L. Gradient descent for this particular problem consists of a sequence of updates

$$K_t = K_{t-1} - \eta\frac{\partial L}{\partial K}(K_{t-1}), \tag{5}$$

where η is the step size.

Discussion. Why does gradient descent "work"? You might want to think about what it means for it to be working.

Implementation. Write a script that does 20 gradient updates and plot the loss values and the sequence $K_1, K_2, \ldots, K_{20}$ against the number of updates.

Discussion. What is a reasonable K_0?

Discussion. What is a reasonable η?

The results of gradient descent should look like Figure 1 and 2. The loss is close to zero after only a few updates.[4]

Discussion. We now have two optimization approaches to solving the same optimization problem. Which one is better, and better in what sense?

[4]Most biochemists these days would use GraphPad Prism `https://www.graphpad.com/scientific-software/prism/prism/` to find these constants.

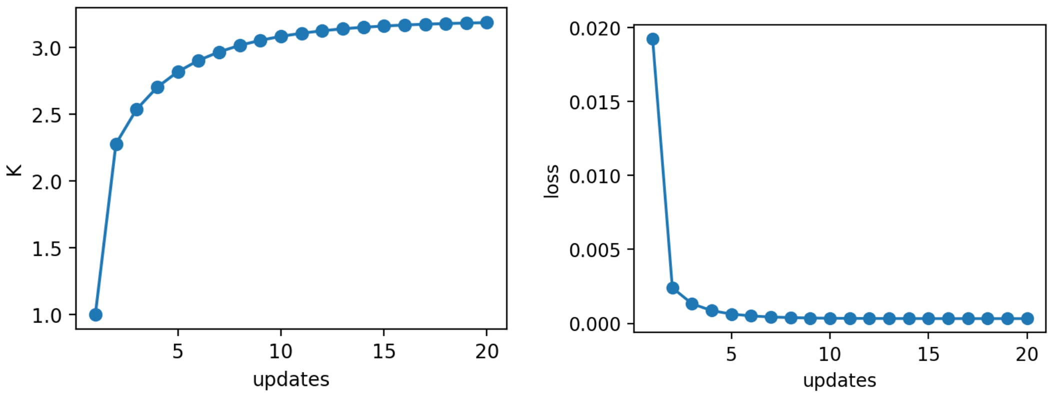

Figure 1: The result of gradient descent when starting from $K = 1$ with $\eta = 50$.

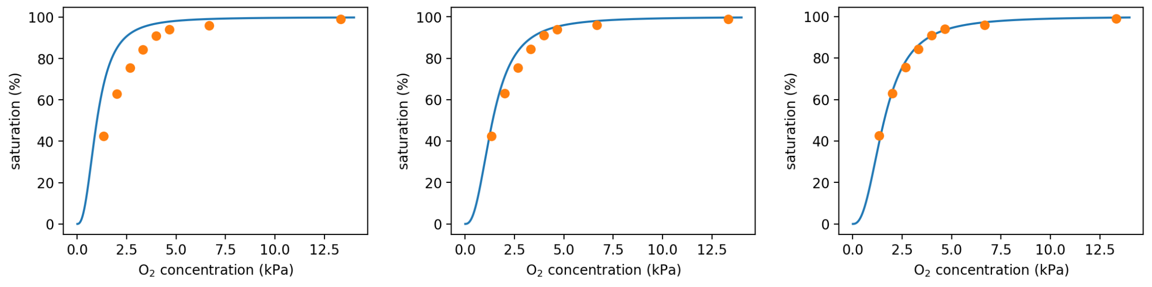

Figure 2: Snapshots of the curves during gradient descent. *Left:* The curve at initialization before gradient descent. *Middle:* The curve after 1 gradient update. *Right:* The final curve after 20 gradient updates.