CS 322 Project 1: Notes on Local Linear Approximations to Functions

1 Linear Fit in One Dimension

Suppose we have a set of points (x_i, y_i) that are samples of some function $y = f(x)$. We think the underlying function may be roughly linear, and so we want to model it with a linear function. Thus, we want to find a function $f_a(x) = mx + b$ that best approximates our sample points. If we have only two sample points, then the best answer is:

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \begin{bmatrix} b \\ m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

If you multiply this out, you will see that it simply says that the function $f_a(x) = mx + b$ passes through the points (x_1, y_1) and (x_2, y_2) – a perfect match! Note the column of ones – this allows us to include the constant term b in our function. We also observe that we need at least two points: if we only have one point, then there are an infinite number of linear functions through that line, and our system is *underdetermined*

Now, if we have more than two sample points, we end up with:

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} b \\ m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

This is of course a least squares problem. Using least squares, we will get m and b, which define the function $f_a(x) = mx + b$ that best approximates all of our sample points.

2 Extending to multiple dimensions

Extending this to more than one dimension is straightforward. Suppose (as in the project) we have 3 dimensions of input (the r, g, and b values) and three functions we want to compute (the best linear fit to the c, m, and y values at that point, respectively). Then we have the following system:

$$\begin{bmatrix} 1 & r_1 & g_1 & b_1 \\ 1 & r_2 & g_2 & b_2 \\ \vdots & & & \vdots \\ 1 & r_n & g_n & b_n \end{bmatrix} \begin{bmatrix} b_c & b_m & b_y \\ a_{rc} & a_{rm} & a_{ry} \\ a_{gc} & a_{gm} & a_{gy} \\ a_{bc} & a_{bm} & a_{by} \end{bmatrix} = \begin{bmatrix} c_1 & m_1 & y_1 \\ c_2 & m_2 & y_2 \\ \vdots & & \\ c_n & m_n & y_n \end{bmatrix}$$

Here, we are finding the three best functions f_c, f_m, and f_y, where $c = f_c(r, g, b) = b_c + a_{rc}r + a_{gc}g + a_{bc}b$ (and likewise for f_m and f_y). Note that we are computing all three functions simultaneously, so our least squares solution will be a $[4 \times 3]$ matrix. In this case, for each function we are searching for the best *plane* that approximates our sample points, and a plane is uniquely determined by 4 points. This leads us to conclude that we need at least 4 points in our system to avoid having an underdetermined system.

Because we want our fit to be local, we only want to perform the least squares fit on nearby points to each grid point. To do this, you should determine which of your sample points are within some specified distance d. Make sure you set d so that you get a reasonable set of points in the interior of your grid – you want to avoid having an underdetermined system, but you also dont want to get all of them. If you wish, you can use the **find()** command to get the sample points that are closest. For each grid point in your local linear fit, once you find the nearest sample points you can construct the above system and solve to to get the linear functions for the CMY value. Then, all you need to do is evaluate each of the three functions at that grid point to get the CMY value. You should perform this process for each grid point in your grid.