Prediction Method by Solving a Linear System

Nathaniel Callens Jr. and Kelly Chang

April 2022

To predict pixels values, we start on the second row, second column and iterate over this inside matrix of the image. Each pixel's prediction is determined by the pixel to its left, top left, directly above, and top right. Each one of these pixels has a known value and we also give them coordinates in the x-y plane so we can fit a line of best fit to them using a system of equations. The hyperplane allows us to make a prediction for the unknown pixel.

The top left point away from the target pixel is given the point (-1,1), the pixel directly above the target pixel is given (0,1), the top right pixel given the value (1,1) and the pixel immediately to the left has coordinates (-1,0). This makes the target pixel's coordinates (0,0).

Using the formula for a line in 3D as $ax + by + c$, the original system of equations has four equations and four unknowns as shown below.

$$-a + b + c = z_0$$

$$b + c = z_1$$

$$a + b + c = z_2$$

$$-a + c = z_3$$

Since we have more equations than unknowns, we will not be able to appropriately create a solution to the system $Ax = b$ where A is

$$A = \begin{bmatrix} -1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

and b is

$$b = \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{bmatrix}$$

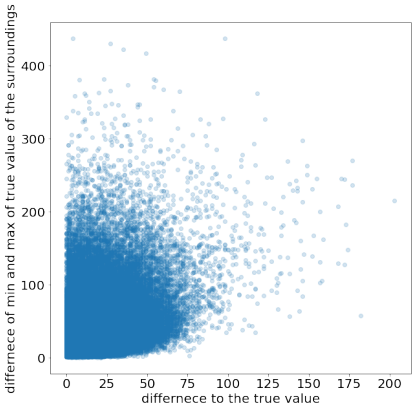
To rectify this, we create a linear combination of the rows to make a system of three equations and three unknowns. The new system is

$$A = \begin{bmatrix} 3 & 0 & -1 \\ 0 & 3 & 3 \\ 1 & -3 & -4 \end{bmatrix}$$

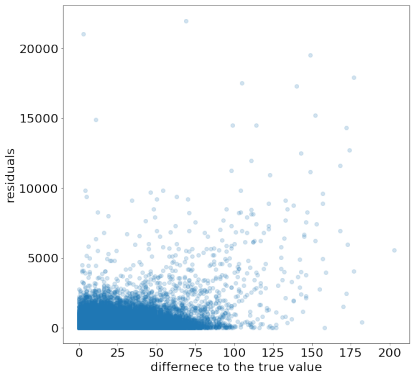
$$b = \begin{bmatrix} -z_0 + z_2 - z_3 \\ z_0 + z_1 + z_2 \\ -z_0 - z_1 - z_2 - z_3 \end{bmatrix}$$

This new system of equations yields a matrix A which is of full rank and can thus be solved using numpy.linalg's built in "solve" method, which accepts a matrix A of full rank and a vector b, and solves for x. This operation is performed for every pixel on the inside portion of the image matrix. Inside portion here refers to every row and column but the first and last. Predictions are generated by taking the value of c from the computed $ax + by + c$ found at each pixel by doing np.linalg.solve on the matrix A and the vector b. The vector b changes with every pixel because it is a combination of surrounding pixels, but A never changes. This gives a unique value c for every step and hence a new prediction for every pixel.

We also measure the difference between the maximal pixel value and minimal pixel value surrounding each target pixel. These get stored as a measure of the gradient. An alternative measurement of gradient that we also use is the distance from each of the four surrounding points to a fitted hyperplane. The hyperplane is fit using the four points and the sum of residuals used as a measure of gradient as well.



(a) Difference of min and max vs. error of prediction.



(b) Residuals vs. error of prediction.

Figure 1: Different measurements vs. predicted error.