STEM

Young

November 4, 2016

1 Linear Regression

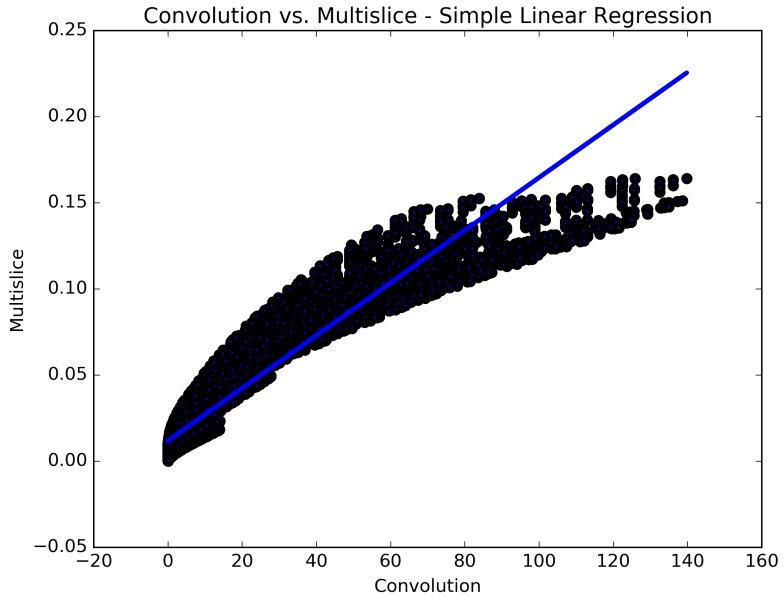
1.1 DONE Simple Linear Regression 10/15/16

$$I.mult = \beta_1 * (I.conv) + \beta_0$$

For the initial model, I reduced the two data sets from 3D to 1D and performed linear regression with a bias term. The least squares estimates for intercept is 0 and slope is 0.0011319.

It is clear that relationship between the two image sets is not linear (see scatter plot with regression line). Also, there seems to be descrete "jumps" in the response variable that the linear model does not capture. I did not try cross-validation with the initial model.

There are many ways to improve the model. First, I would build a model that imposes a restriction on the response variable. One example would be logistic transformation which will output in the range of [0,1]. We could then rescale the output to match the range of the Multislice image intensities.



Secondly, I would develop a feature set that has more explantory power. One natural way to start would be to include the dimensional indices as features to account for spatial correlation. However, this could be complicated as there seems to be clusters of high intensity pixels in the center and corners of the image. Either a kernel method or K-means method might be useful to account for spatially-clustered pixel intensities. Alternatively, we could perform segmentation on the different regions.

Third, some form of piecewise model (i.e. Decision Trees) or locally-weighted models might better account for "jumps".

- Python files: olr.py

1.2 Multiple Regression 10/30/16

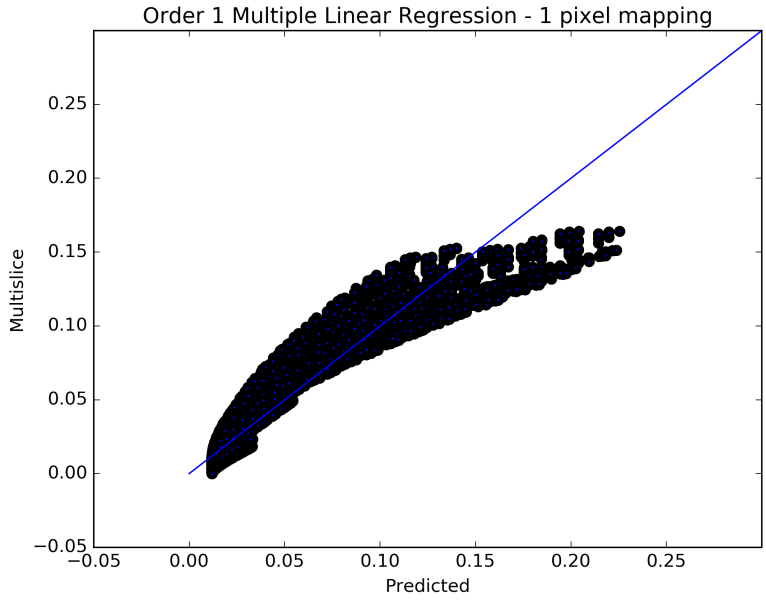
For each layer, I generated a .png file with side by side images of convolution, multislice, and their difference (after normalization). These files are in data/images/ directory. It looks like multislice images tend to be smoother than convolution images.

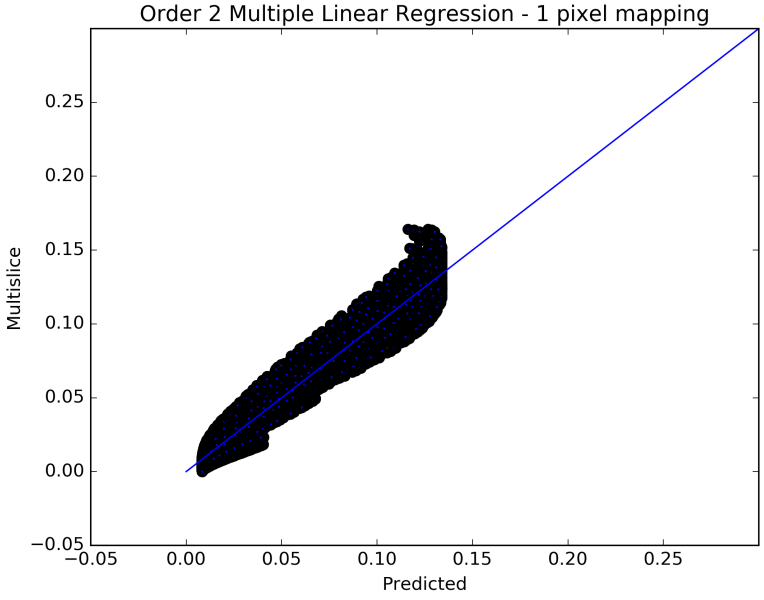
I fit various multiple linear regression models using

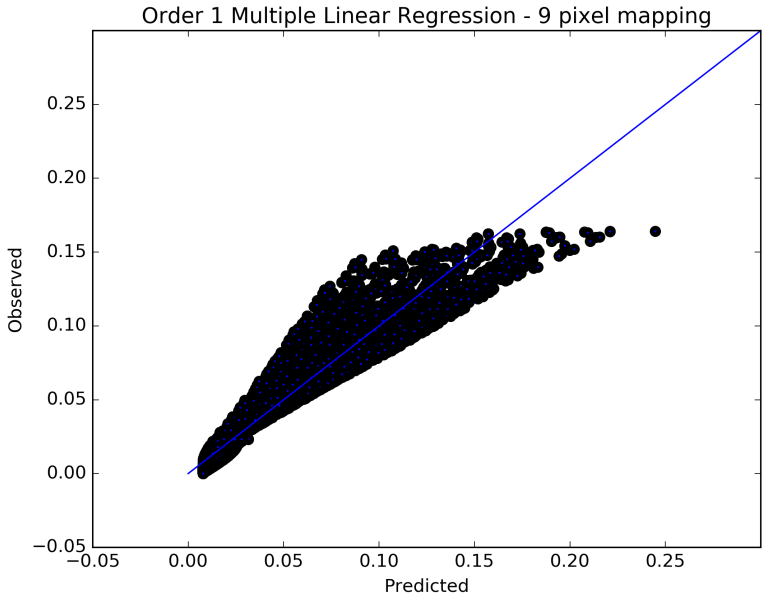
- 1 pixel or 9 pixels (pixel and 8 neighbors)
- gradient direction/magnitude
- linear and quadratic predictors

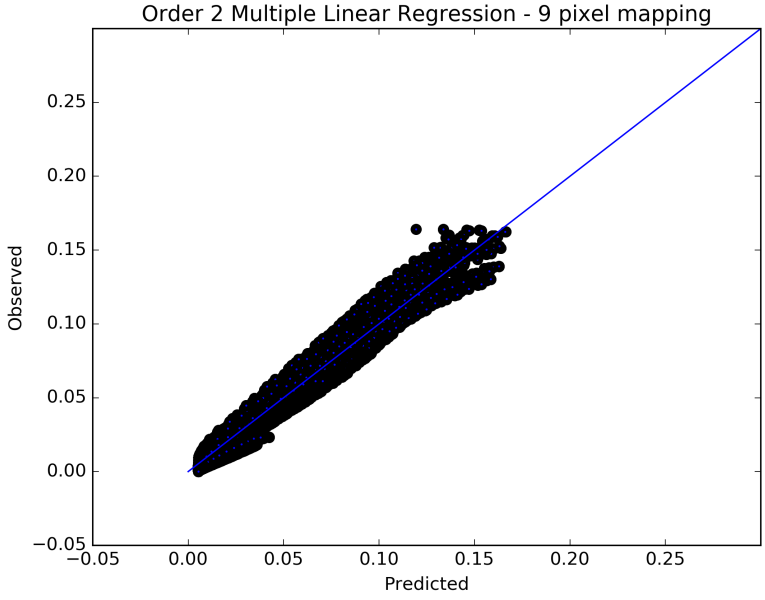
The graphs below show the predicted outcome vs. observed (multiscale) intensity as a measure of model fit. It's clear that adding the quadratic term improves the model fit significantly. Gradient information doesn't seem to improve the model beyond the quadratic terms of the neighbor intensities.

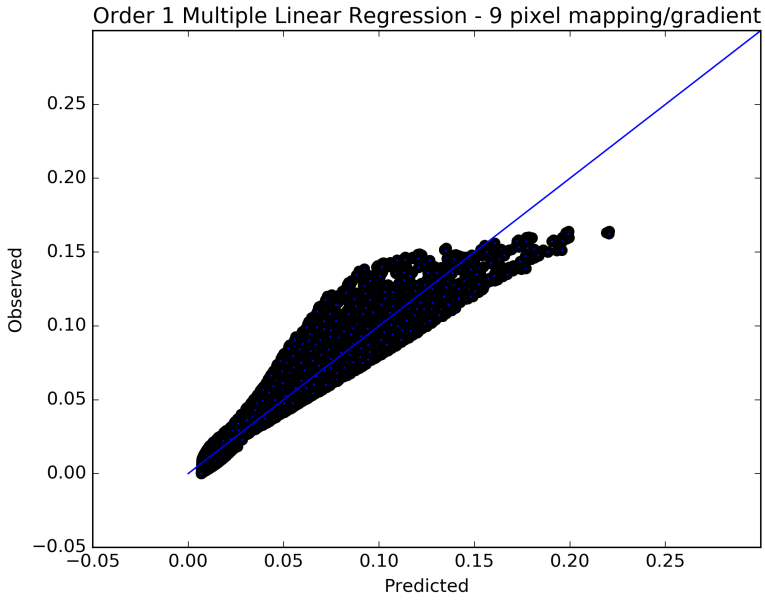
I plan to perform cross-validation to compare these models. I also would like to experiment with applying smoothing filters to the convolution image.

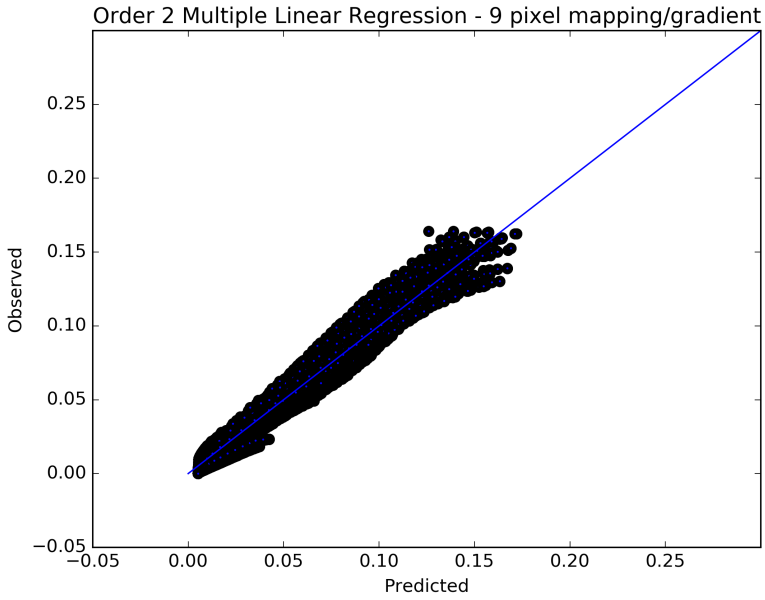

Order 1 Multiple Linear Regression - 1 pixel mapping
Multislice
Predicted
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.05
0.00
0.05
0.10
0.15
0.20
0.25


Order 2 Multiple Linear Regression - 1 pixel mapping
Multislice
Predicted
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.05
0.00
0.05
0.10
0.15
0.20
0.25


Order 1 Multiple Linear Regression - 9 pixel mapping
Observed
Predicted
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.05
0.00
0.05
0.10
0.15
0.20
0.25


Order 2 Multiple Linear Regression - 9 pixel mapping
Observed
Predicted
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.05
0.00
0.05
0.10
0.15
0.20
0.25





- Python files: olr2.py

2 Data

2.0.1 Convolution

- 50 by 36 by 20
- intensities = [0, 139.7768]

2.0.2 Multislice

- 50 by 36 by 20
- intensities = [0, 0.1641]