UCLA College of Letters and Sciences
Department of Mathematics

MATH 170B PROJECT 1 REPORT

Artem Pashchinskiy and Jianzhen Wang

June 11, 2018

In most treatments of Gaussian linear models, constant variance of error is assumed under the ordinary least squares approach. However, in many real world applications, noise variance tends to be correlated with the input variable(s). In this project, we try to estimate the parameters of data-generating process in 5 datasets with an unknown distribution of noise.

In the problem specification, we are told that the linear model is of the form

$$y_i = a * x_i + b + e_i$$

We started our work with creating scatter plots of the provided data. Based on the estimated distribution of variance, we concluded that heteroskedasticity is present in all 5 sets of data. That means that the variability of the response variable is unequal across the range of values of the independent variable. In order to find the best estimation for the parameters a and b in each data set, we formulated five different models, including Weighted Least Square Regression, Huber Regression, Ridge Regression, LASSO, and Generalized Least Square Regression to accommodate the volatile nature of the independent noise e_i.

From a technical realization standpoint, we used Python coding language along with functions from publicly available packages Pandas, NumPy (for data processing) and SciPy Sklearn (for regression models). Full implementation of the code can be accessed on the Git Hub repository.

Ordinary Least Squares Regression

The first and the most straight-forward approach was to perform Ordinary Least Squares Regression. Although OLSE estimators are unbiased, it does not accomodate heteroskedasticity. Hence, we did not expect this model to have the most optimal performance. We rather use it as a means to develop appropriate data transformations for more complicated methods, such as Weighted and Generalized Least Square regressions.

Huber Regression

Since we observed some potential outliers, we also implemented the Huber Regression on the given data. Huber Regression takes into consideration impacts that the outliers make on the model, but it also makes sure that the loss function is not heavily influenced by the outliers. With Huber Regression, we optimized the squared loss and the absolute loss for the samples, in order to estimate a and b for the best-fitted line.

Ridge Regression

We utilized the ridge function in provided in Python sklearn library to fit the data. We made sure that the function performs the cross validation procedure by iteratively splitting provided data into training and testing samples. We ranged values of λ constant in the loss function

$$||Y - X\vec{a}||^2 + \lambda||\vec{a}||^2$$

from -20 to 20 with 0.5 increments to find the best fitting model model. With the best alpha value for the loss function, we estimated the parameters $\vec{a} = (a, b)$ for the fitted regression line.

LASSO

We applied the underlying LASSO cross validation function in Python on the response variable y_i with the 0.0001 length of path and 1000 values of λ along the path. We then used the loss function to estimate the parameters a and b, trying to fit a regression line on each given data set.

Although all models presented above have their advantages over the Ordinary Least Squares regression, none of them is specifically targeted to address the heteroskedastic nature of the data. Hence, we believe that results of the next three models are more reliable for this specific scenario.

Weighted Least Squares

One of the ways to approach x-dependent distribution of noise is to scale the contribution of various data points to the values of estimated coefficients. One of the simple dependencies that we were able to come up with based on the visual observations of data is

$$var(e_i) = \sigma_i^2 = \sigma^2 x_i$$

If the error of the data-generating process is following this dependency, we can use

$$\sqrt{x_i}$$

as a weight coefficient of error, i.e. transform the data into a new set that has homoskedastic errors.

Although Python packages have built-in functions for an automatic computation of weighted least square regression, we decided to stay conservative and perform the data transformation manually and use built-in functions only for applying OLSE to the transformed homoskedastic model.

Generalized Least Squares - I

Although in many cases Weighted Lest Squares gives a good estimate of the true coefficients a and b, it is heavily dependent on how well the error dependency was estimated. Since we had limited amount of information about the error, we decided to also implement a more general framework of Generalized Least Squares regression.

According to the assumptions of Generalized Least Square Regression, we know that the distribution of the dependent variable y_i given x_i is normal, but the variance of the residual is not constant.

$$(y_i \,|\, x_i) \sim \mathcal{N}(a * x_i + b, var(e_i|x_i))$$

Advantage of the GLSE method lies on the fact that opposed to WLSE it allows to automatically deduce the coefficients of error dependency. In this case we specified our model as

$$var(e_i|x_i) = \alpha * x^{\beta} \tag{$\star$}$$

Since squares of OLSE residuals σ_i can be used as estimates of true errors e_i, we can apply OLSE to the model

$$\ln(\sigma_i^2) = \alpha + \beta \ln(|x|)$$

to get the estimated values $\hat{\alpha}$ and $\hat{\beta}$. Using this estimates, we can predict

$$var(e_i|x_i) = \widehat{\sigma_i^2} = \exp \hat{\alpha} + \hat{\beta} \ln(|x|).$$

Once we have the predicted values, we can weigh the errors similarly to what we did in WLSE method and get a homoskedastic model:

$$y_i^* = az_1 + bz_2 = a_1 * \frac{1}{\hat{\sigma_i}} + b * \frac{x_i}{\hat{\sigma_i}}$$

Finally, we can apply OLSE model to regress y^* against z_1 and z_2 and get the estimations $\hat{a}$ and $\hat{b}$.

Generalized Least Squares - II

Towards the end of project development, we discovered additional input specifying that functional form

$$var(e_i|x_i) = c + dx$$

was used to create noise in the data-generating process. To accommodate this extra assumption, we repeated the of Generalized Least Squares method again using this new error form in the place of assumption ($\star$)

Since this last attempt incorporates both the known methodology of dealing with heteroskedastic error and the the largest available set of reliable assumptions (both a known model for data generating process and a known model for noise-generating process), we assumed that parameter estimates produced by this model should be the most accurate out of all considered models.

Results and Discussion

After obtaining the point estimates for all the models above, we utilized the coefficient of determination (R^2) and the root mean square error ($RMSE$) to assessment the quality of the parameter estimates.

R^2 is a statistical measure of the percentage of the response variable variation that is explained by a linear model. For each set of data, we applied the different models and we calculated the corresponding R^2. The following table summarizes our findings.

reg / dataset	1	2	3	4	5
LSE	0.1639863122	0.3980894410	0.1031999500	0.0275947359	0.2805757920
GLSEI	0.1166125126	0.0915335288	-0.015537303	-0.1297663105	-0.1286472674
GLSEII	0.0464062676	0.0694021845	-0.041146286	-0.1472103778	0.2805757920
WLSE	0.1403769574	0.1834023549	0.0162476341	-0.0178163048	-0.0406054285
LASSO	0.1639739902	0.3980894371	0.1031952438	0.02759473565	0.2805757898
RIDGE	0.1638634355	0.3980877000	0.1031988620	0.02670308780	0.2805165134
HUBER	0.1573759139	0.3937864654	0.0830787080	0.00603328964	0.2581856714

Table 1. R^2 values

Although R^2 can be easily applied to measure the goodness of fit of a linear model, it has some limitations. First, R^2 does not determine if the predictions are unbiased. Based on the residual plots of the given data, our sample is biased, since the independent noise varies with the input variable. Second, R^2 does not indicate the accuracy of a regression model. In other words, high R^2 does not necessarily mean that the model fit the data well, and vice versa for the low R^2. Thus, we tried another metric, RMSE, to access the quality of our models. As its name suggests, the root mean square error is the square root of the average of squared differences between the predicted values and the sample values. The following table shows the RMSE value for each dataset with various regression models applied.

reg/dataset	1	2	3	4	5
LSE	5.2949701	15.4237819	17.5180183	5.0732698	11.1023251
GLSEI	5.4429261	18.9486967	18.6416825	5.4683791	13.9059458
GLSEII	5.6550768	19.1781146	18.8752644	5.5104345	11.1023251
WLSE	5.3692155	17.9650717	18.3476337	5.1903783	13.3525576
LASSO	5.2950091	15.4237819	17.5180643	5.0732698	11.1023251
HUBER	5.2953592	15.4238042	17.5180290	5.0755952	11.1027825
RIDGE	5.3158626	15.4788149	17.7134516	5.1292070	11.2737662

Table 2. $RMSE$ values

Overall, RMSE is the most reliable metric to qualify the goodness-of-fit of a model because it remains unbiased in both cases of homoskedasticity and heteroskedasticity. From the values calculated above, we observed that OLSE actually gives the smallest error for all five datasets.

From this, we conclude that the variance of the noise should be closer to constant (as assumed in OLSE) than to the input-dependent assumptions of GLSE. Although we visually observed some apparent dependency on the plots of residuals, it turns out that none of the functional forms we tried turned out to produce a better result than a simple constant assumption.

Among the models we tested in this project, we concluded that GLSE should give the most accurate estimation parameters for a and b based on methodology available in the literature and our assumptions for the variance functional form. Yet, numerical metrics provided evidence that for these datasets modeling noise variance as a constant provides better results (smaller errors).