**Part 1 Polynomial function**

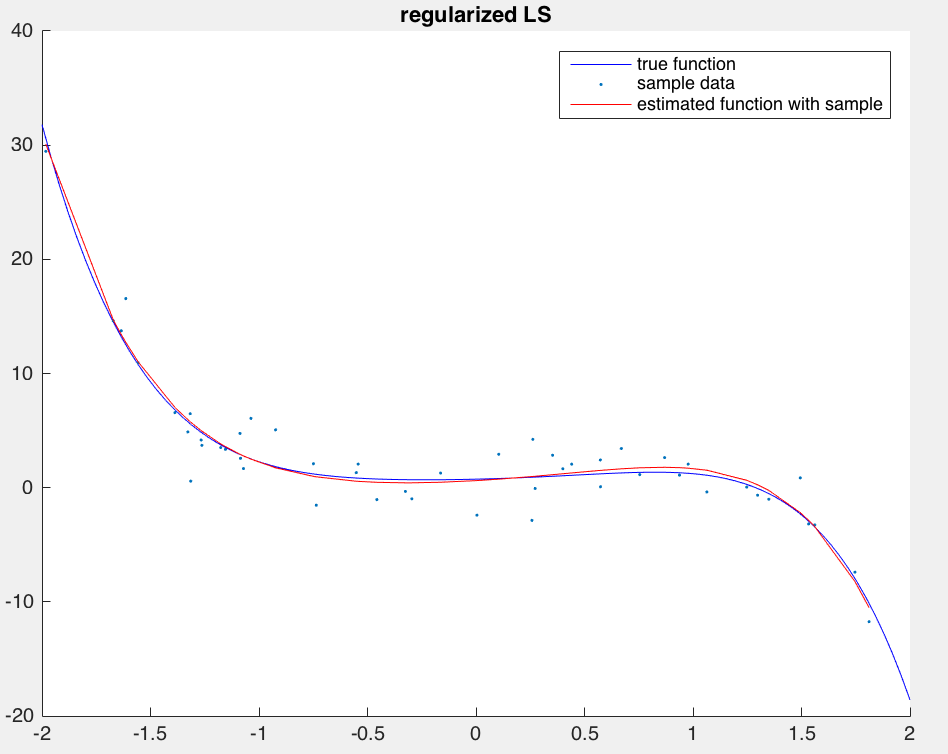
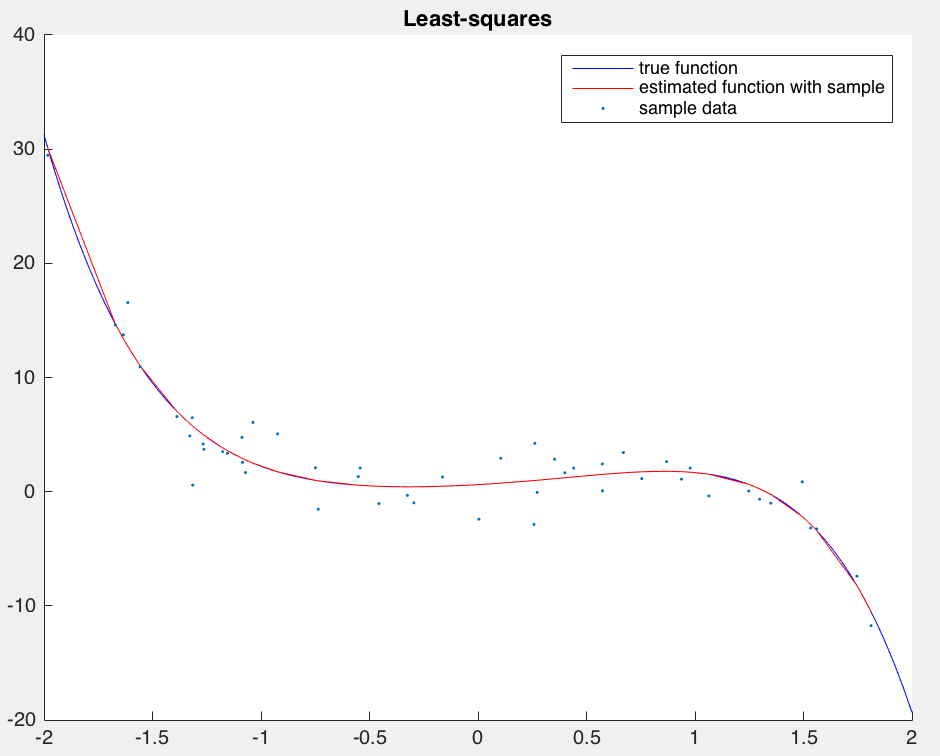
(a) Implement the above 5 regression algorithms for the K-th order polynomial given in (2). In the next problem, you will use these regression methods with a dierent feature transforma-

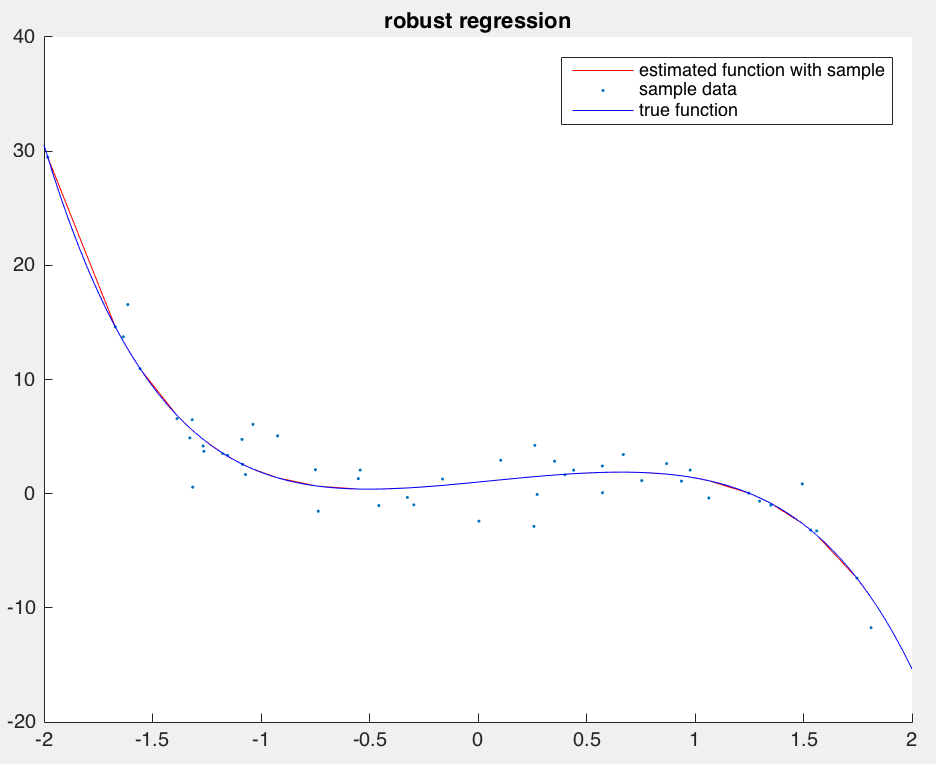
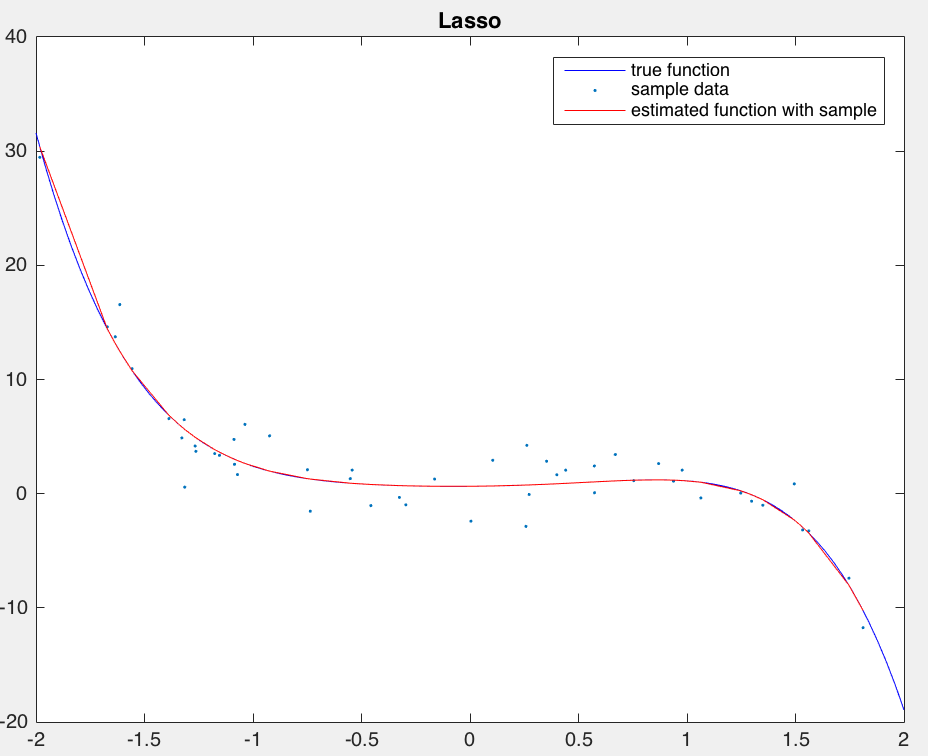
tion (x). Hence, it would be better to separate the regression algorithm and the feature

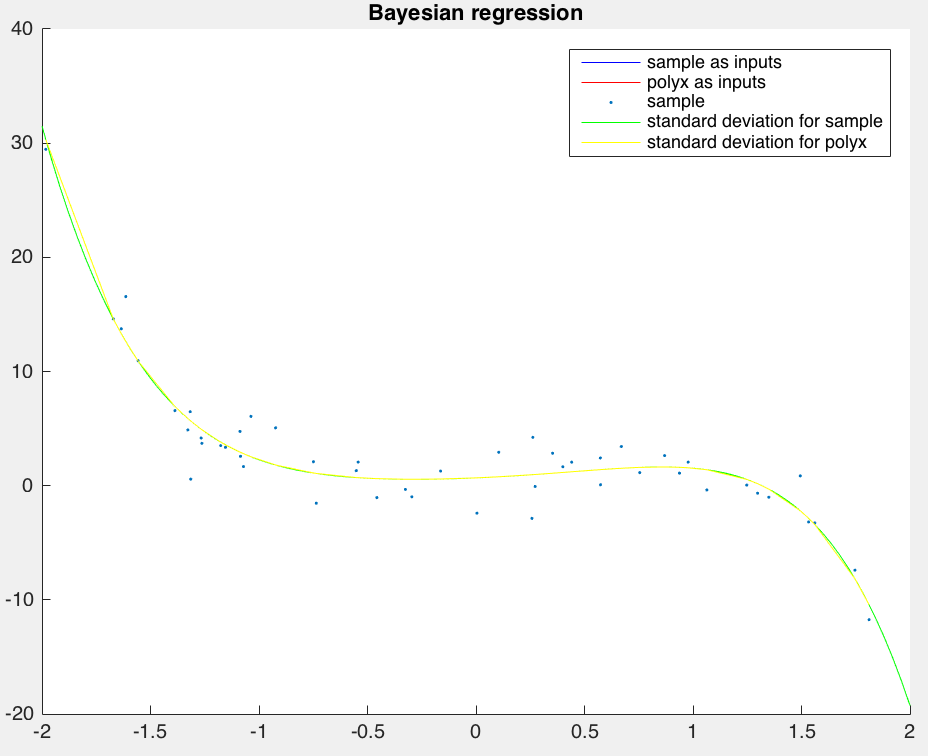
transformation in your implementation.

(b) For each regression method, use the sample data (sampx, sampy) to estimate the parameters of a 5th order polynomial function. Make a plot of the estimated function using polyx as inputs, along with the sample data. For BR, also plot the standard deviation around the mean. What is the mean-squared error between the learned function outputs and the true function outputs (polyy), averaged over all input values in polyx? For algorithms with hyperparameters, select some values that tend to work well.

Answer:







Mean-squared error

Least-squares: 0.4086

Regularized LS: 0.4592

 L1-regularized LS: 0.5698

 robust regression: 0.7680

 Bayesian regression: 0.4086

(c) Repeat (b), but reduce the amount of training data available, by selecting a subset of the

samples (e.g., 10%, 25%, 50%, 75%). Plot the estimated functions. Which models are more

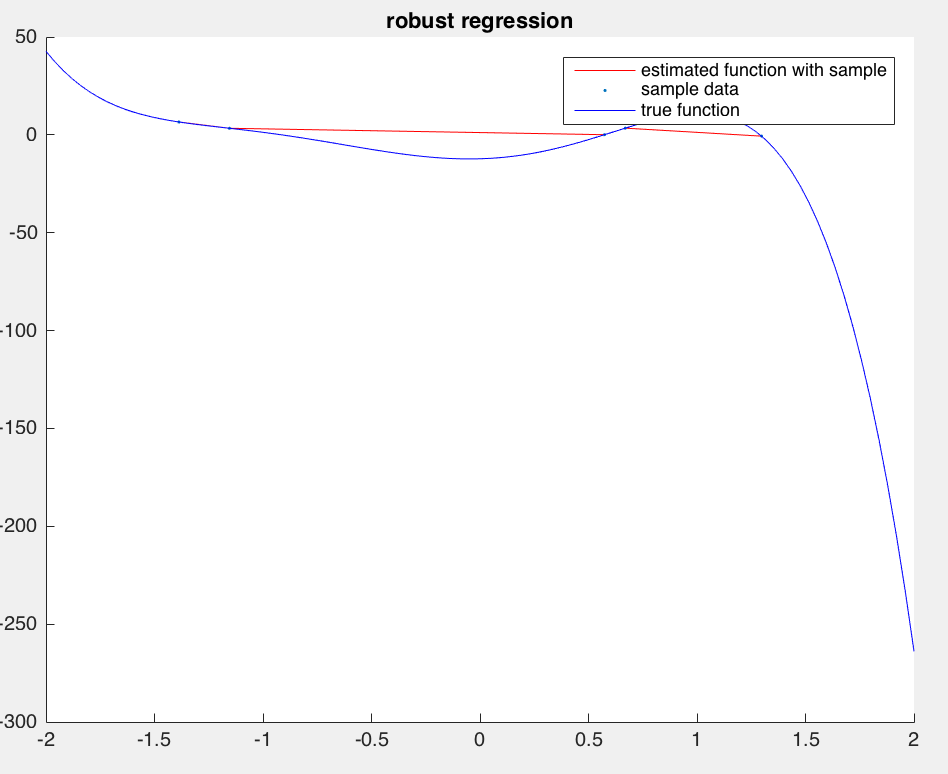
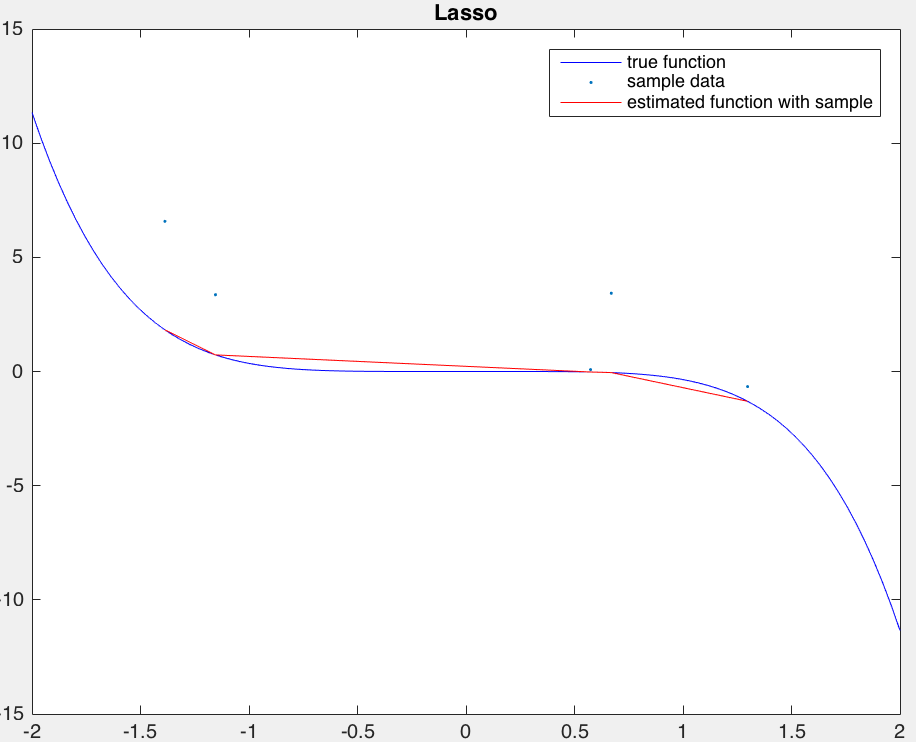
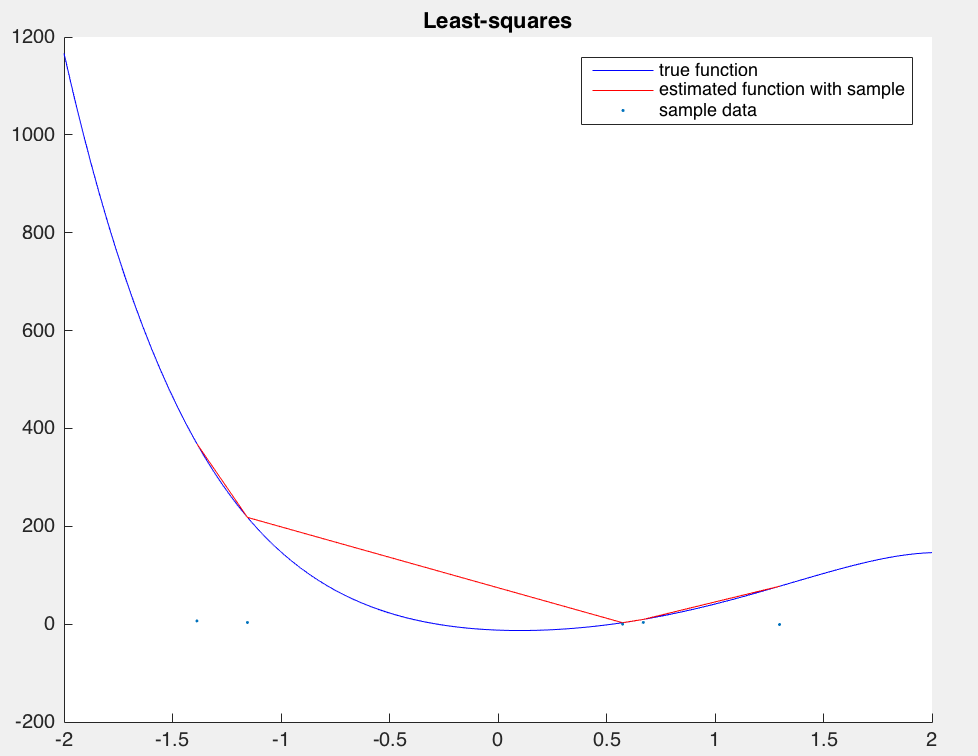
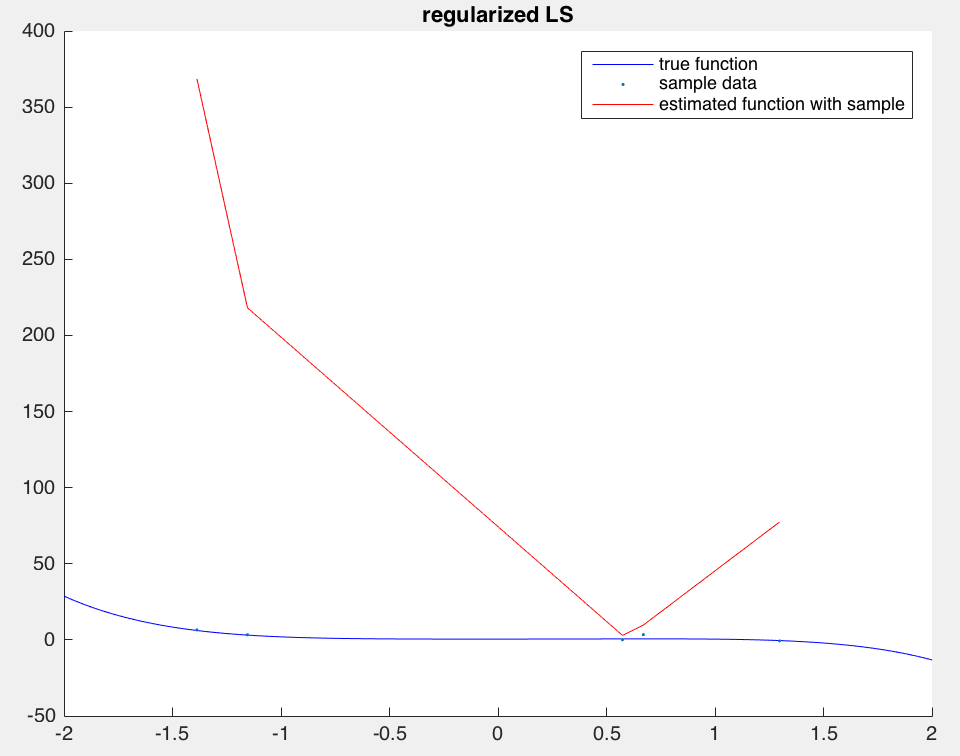
robust with less data, and which tend to overfit? Make a plot of error versus training size, and

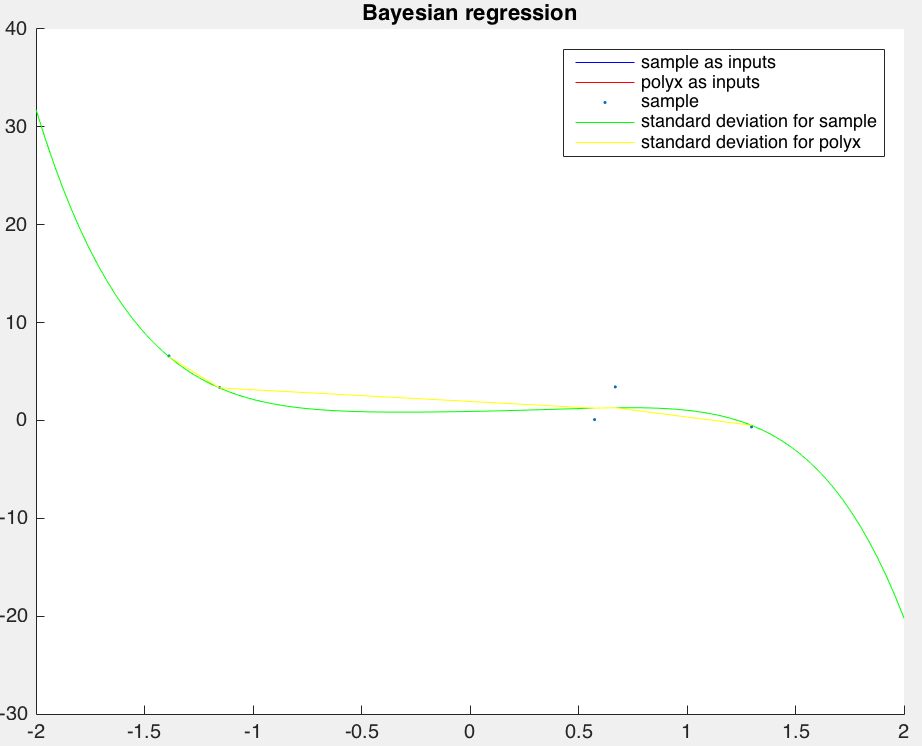
comment on any important trends and findings. (You should run multiple trials with different

random subsets, and take the average error).

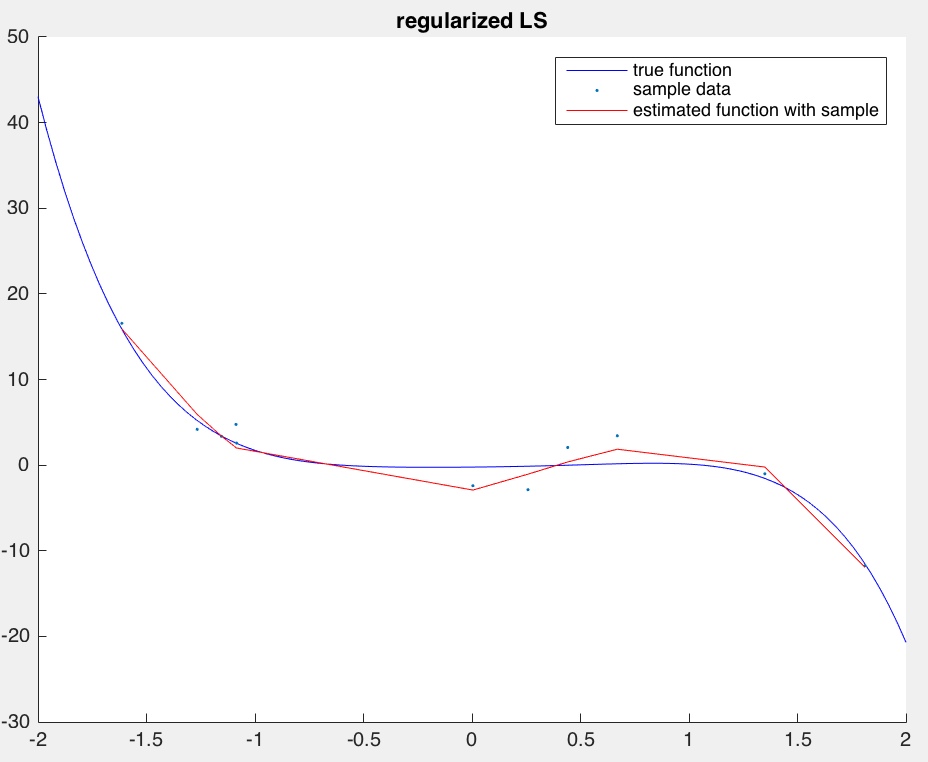
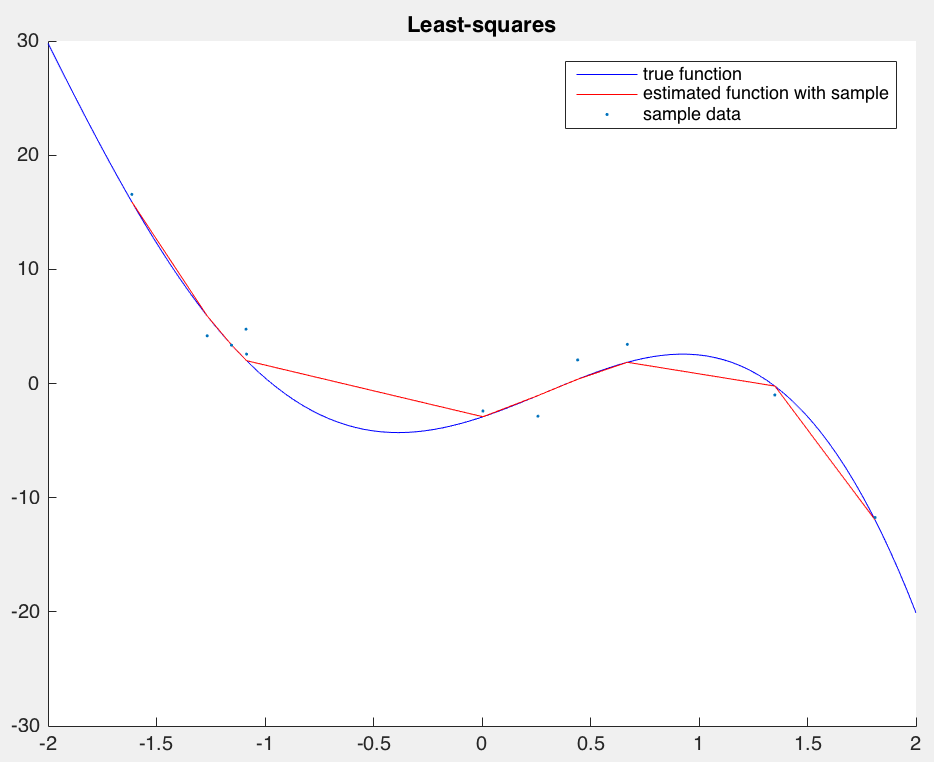
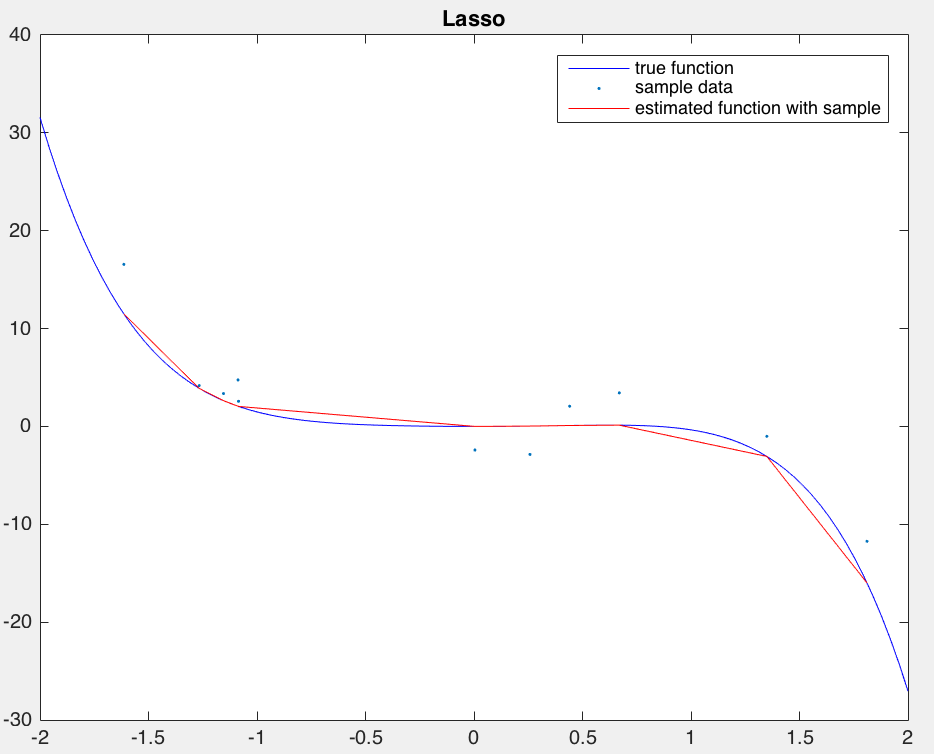
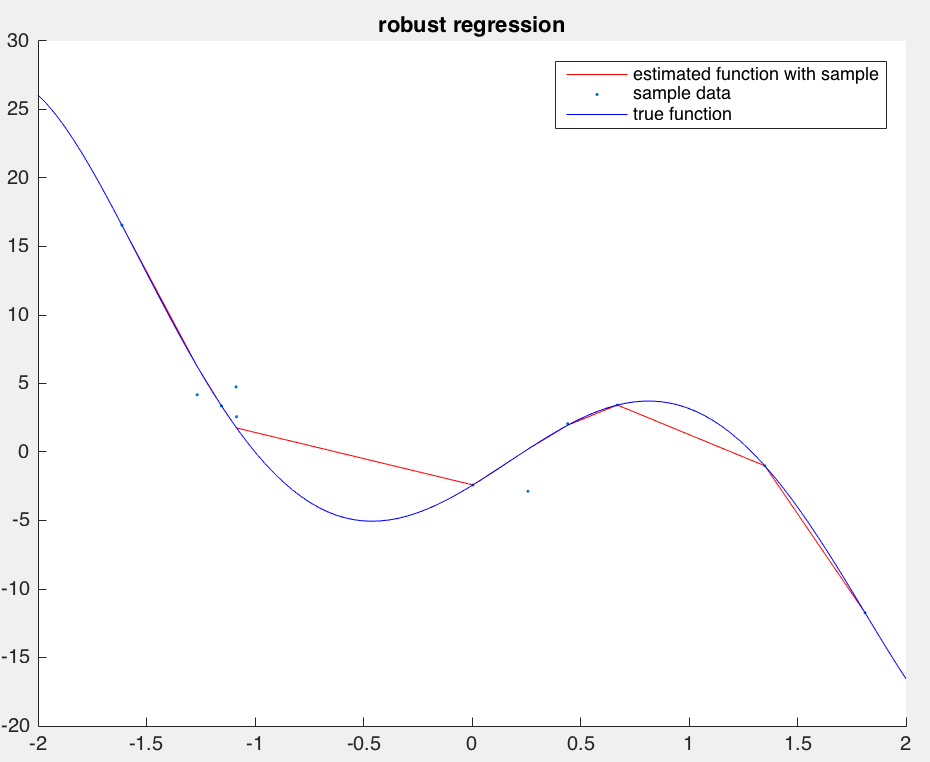
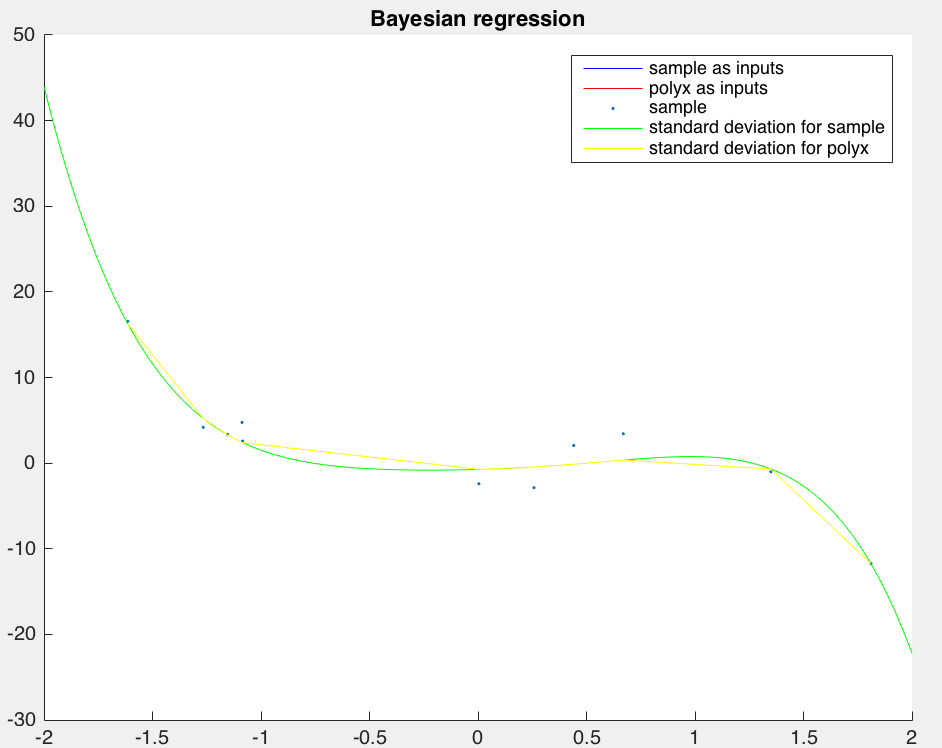
Answer:

10% samples:

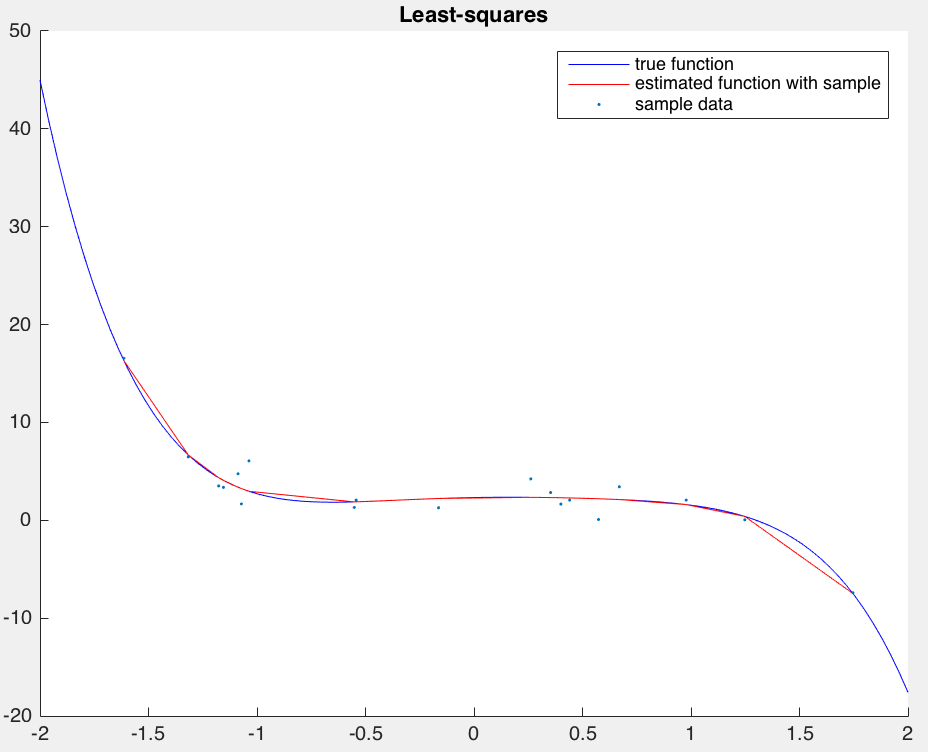
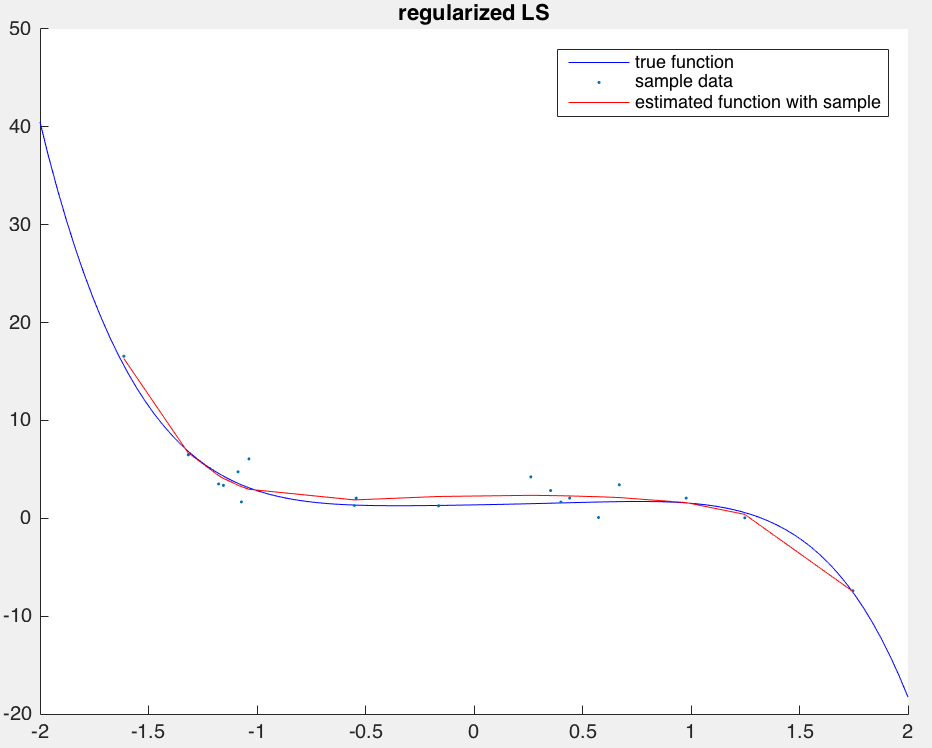
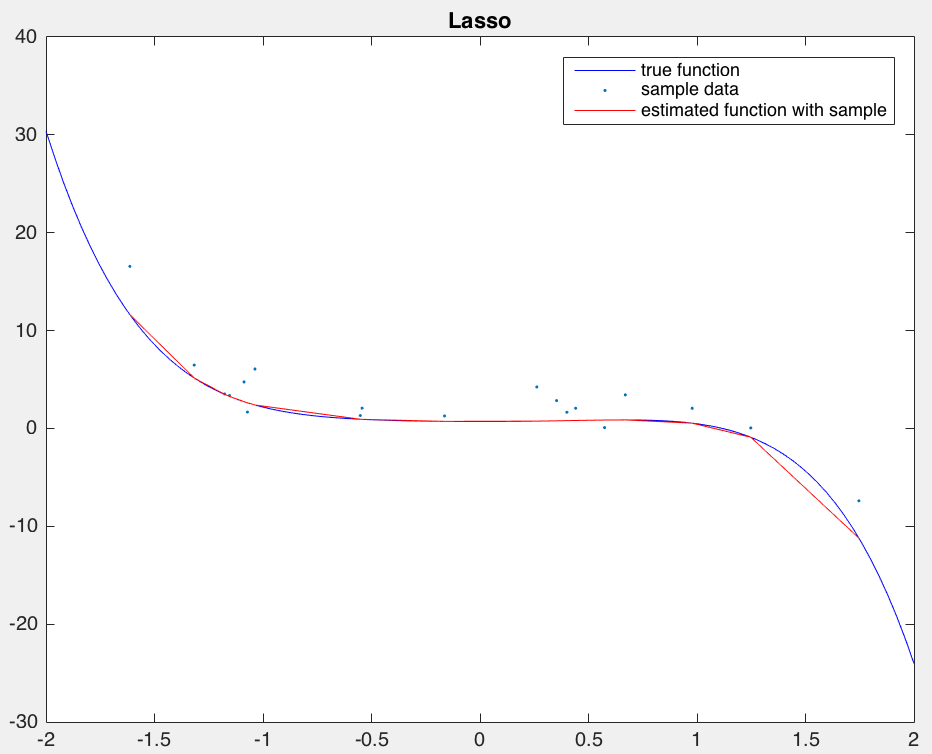
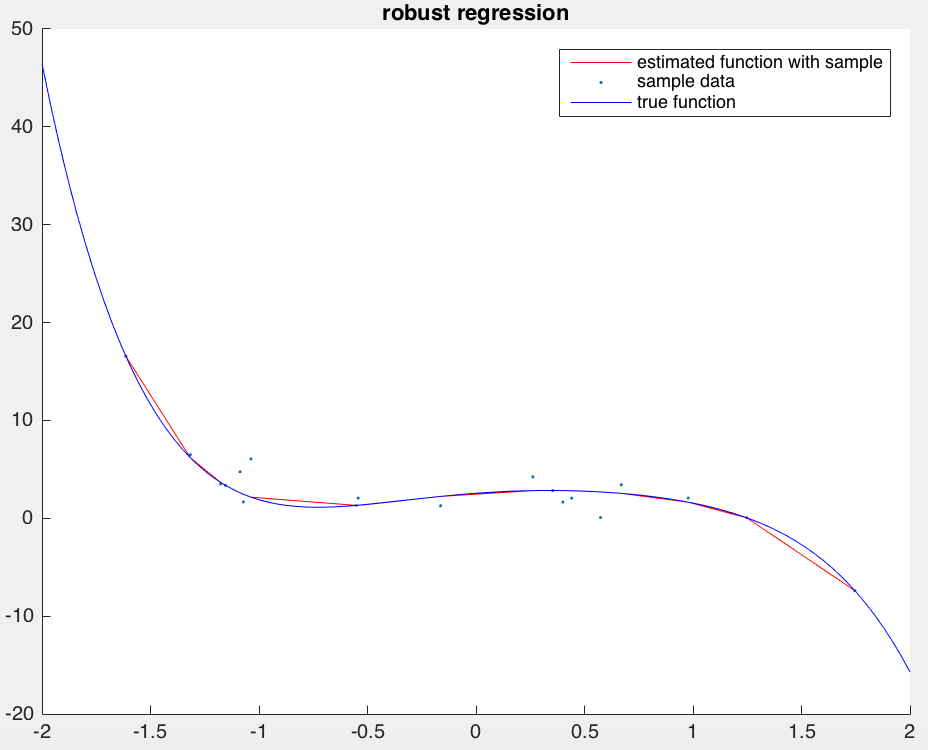
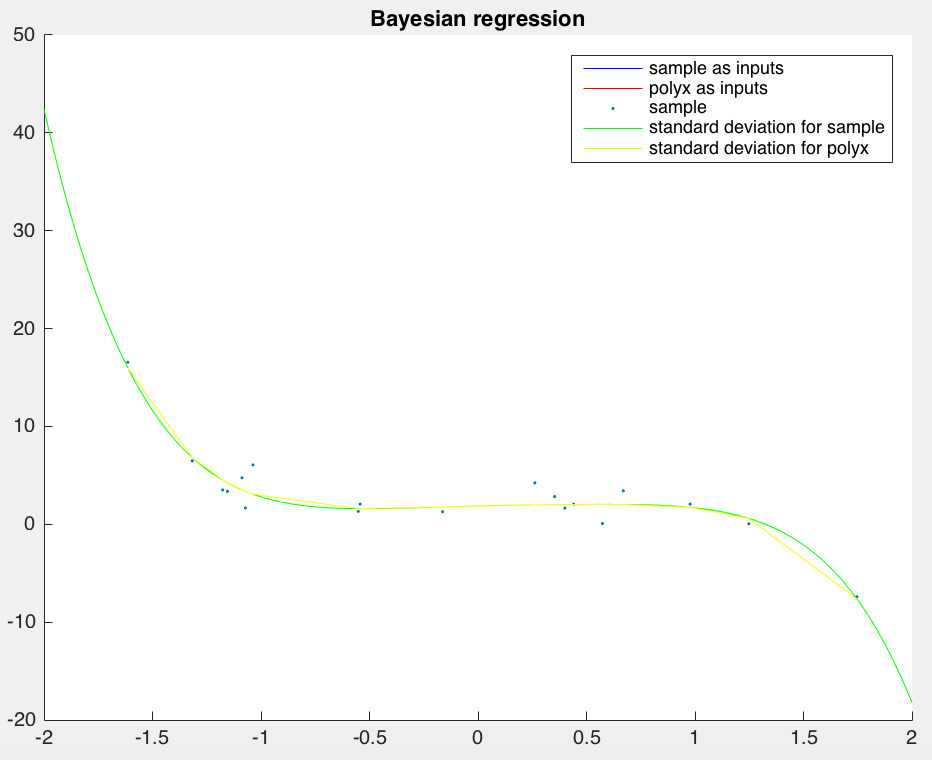




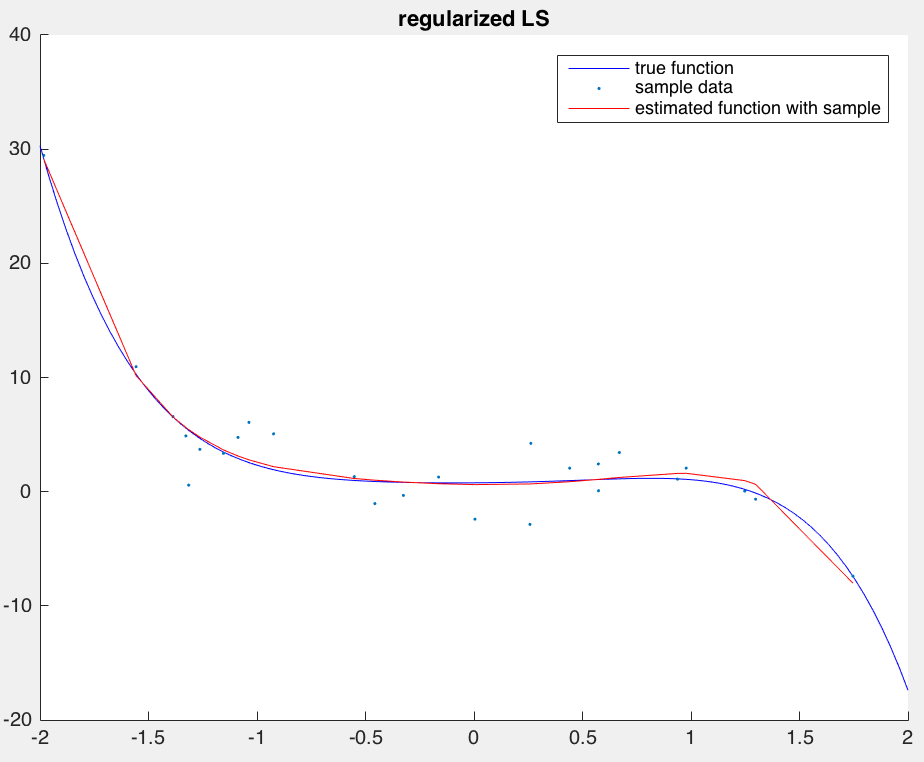
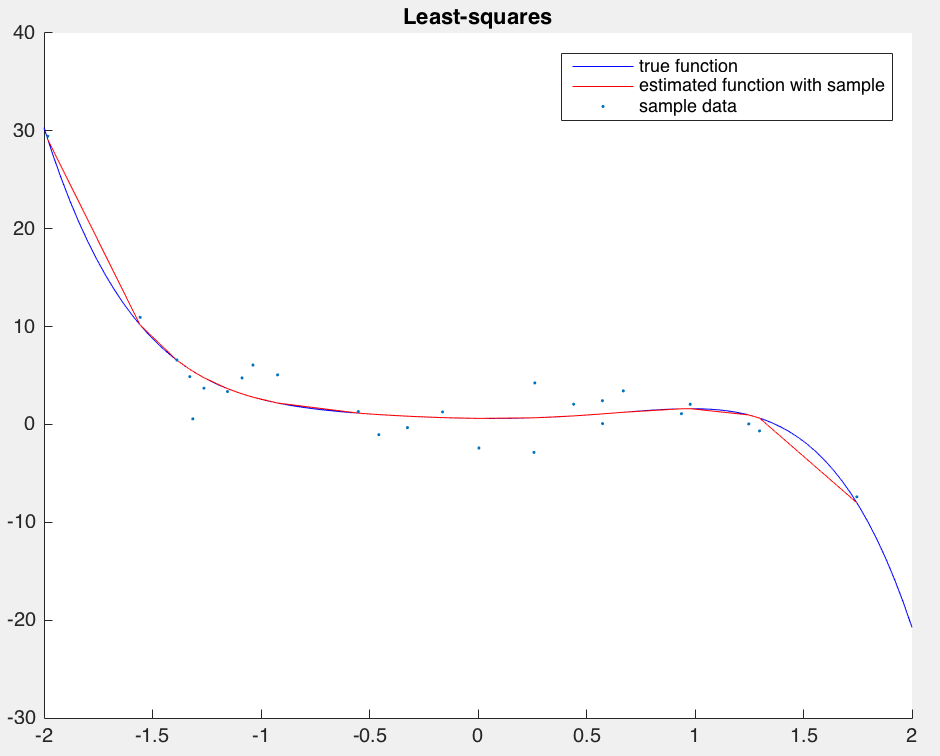
25% samples:

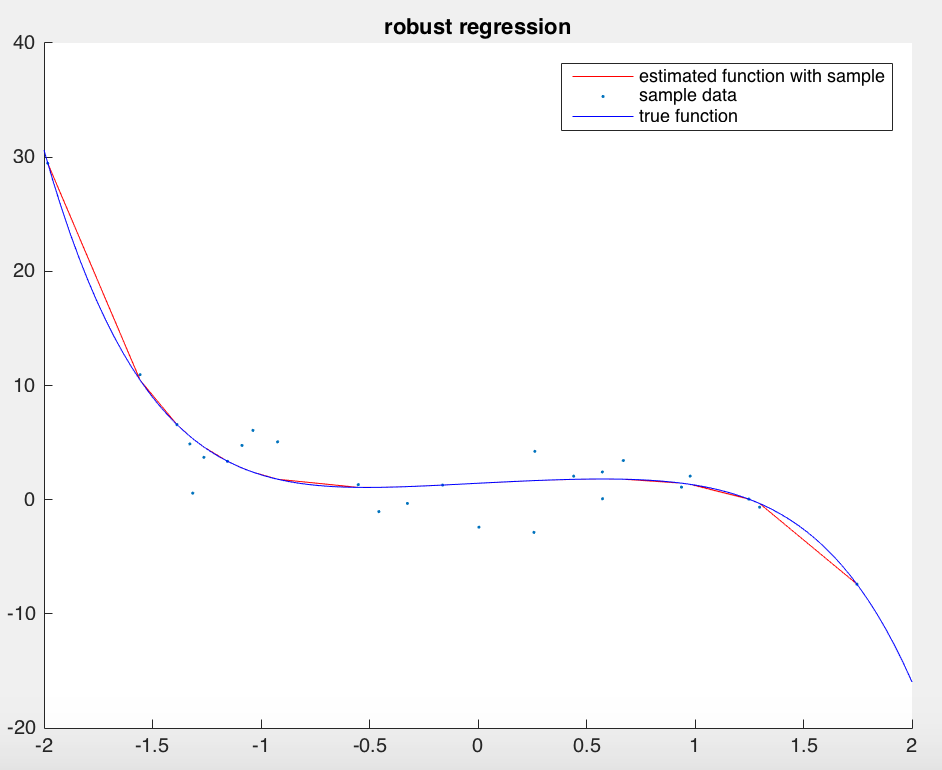
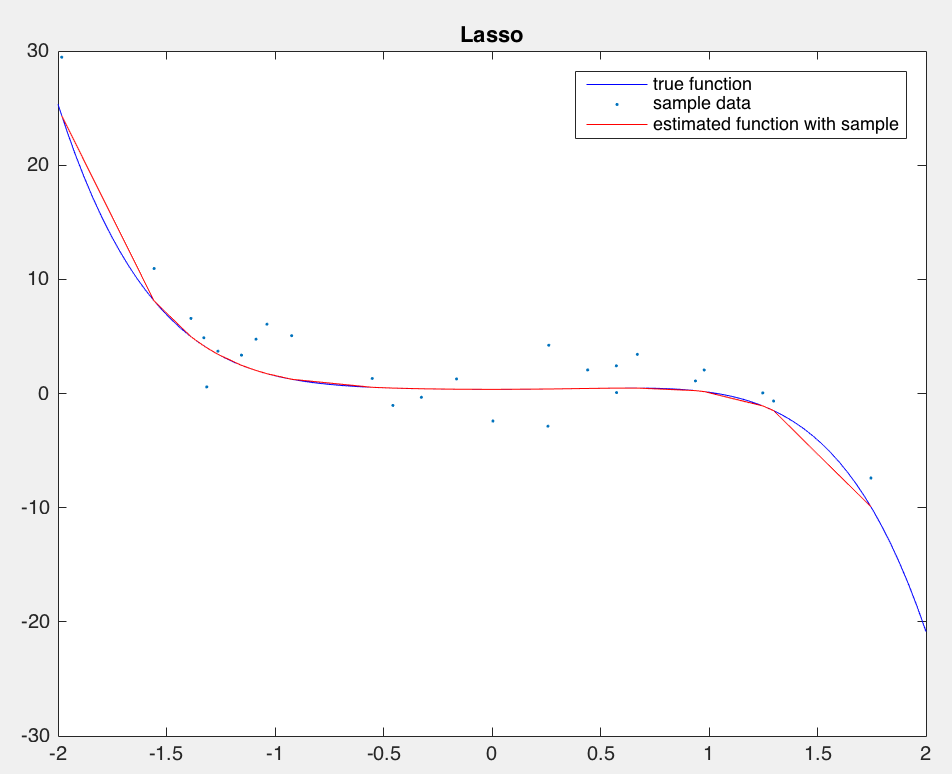
   

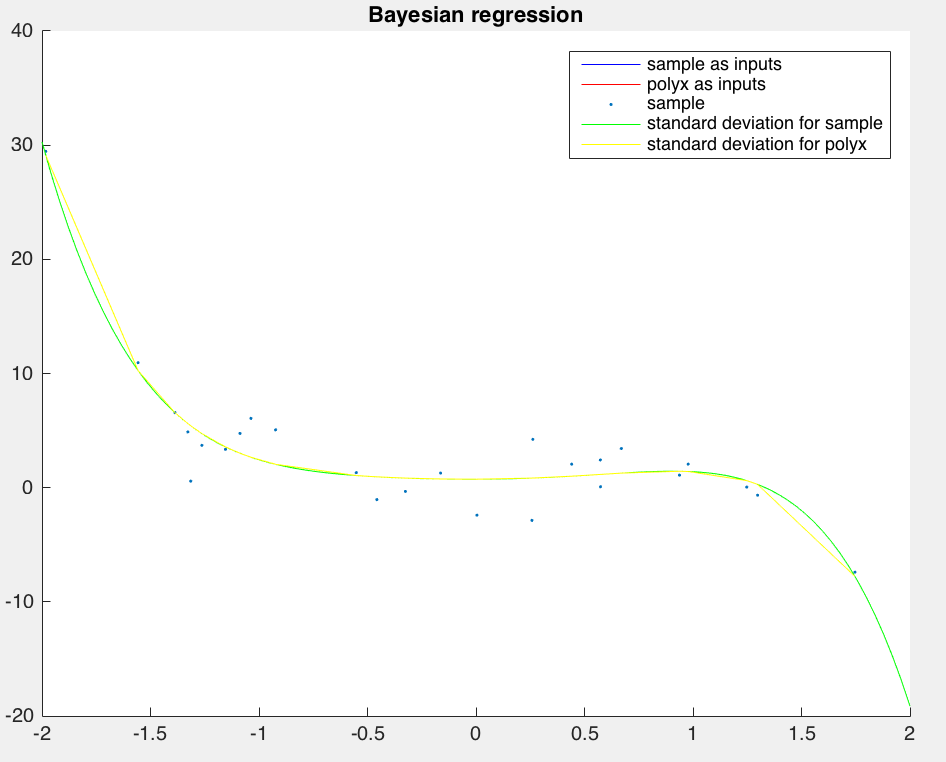
50% samples:

75% samples:





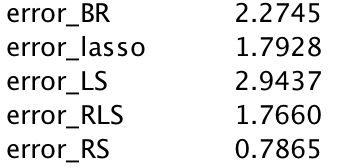


From the result, we can draw a conclusion that Bayesian regression tend to be more robust with less data than other regression method. Regularized LS tend to overfit.

(d) Add some outliers output values (e.g., add large numbers to a few values in sampy), and repeat (b). Which methods are robust to the presence of outliers, and which are most sensitive? Why?

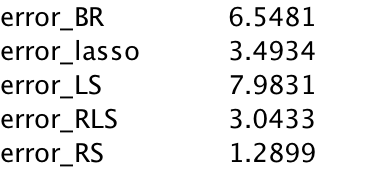
Answer:

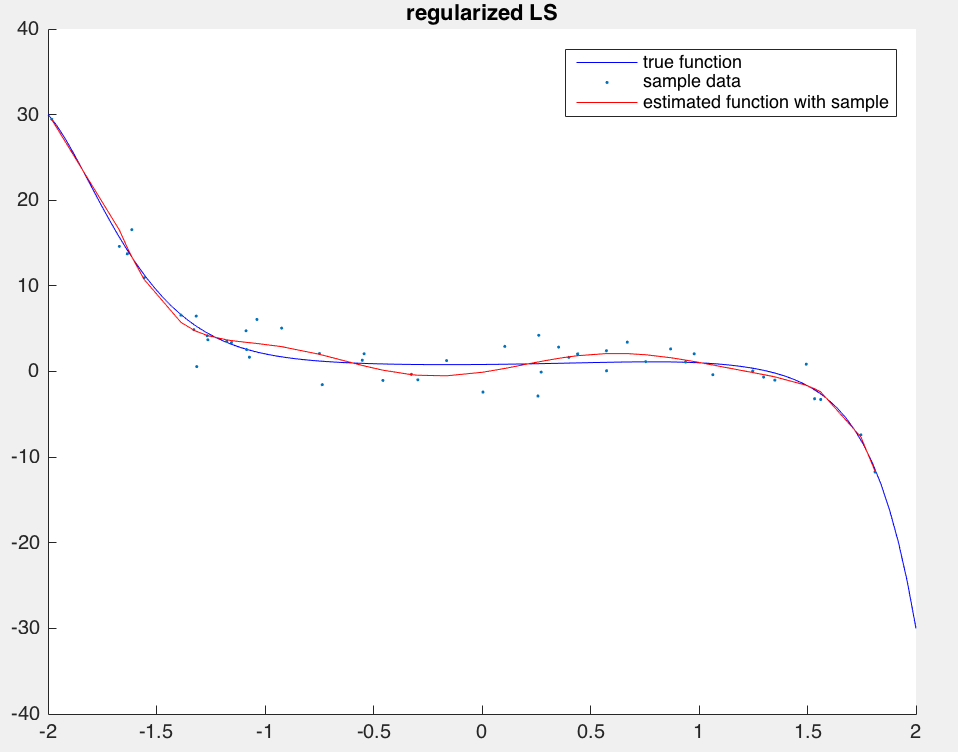
After add some outliers output values, robust regression are more robust to the presence of outliers, and least-square algorithm are most sensitive. Since robust regression methods are designed to be not overly affected by violations of assumptions by the underlying data-generating process. This least-square formulation considers only residuals in the dependent variable.



(e) Repeat (b) but estimate a higher-order polynomial (e.g., 10th order). Which models tend to overfit the data when learning a more complex model, and which models do not? Verify your observations by examining the estimated parameter values.

Answer:





For a higher-order polynomial, regularized LS seems tend to overfit the data when learning a more complex model.

**Part 2 A real world regression problem - counting people**

(a) Let's first look at using the features directly, i.e., set (x) = x. Use the training set (trainx,

trainy), estimate a function with some of the regression algorithms above. Use the test set

inputs testx to predict the output, and compare the predictions to the true output testy. (You

can round the function outputs so they are counting numbers). Calculate the mean-absolute

error and mean-squared error. Which method works the best? Plot the test predictions and

the true counts, and discuss any interesting findings.

Answer:

mean-absolute error

Least-squares: 1.3584

Regularized LS: 1.3584

L1-regularized LS: 1.2565

robust regression: 1.4886

Bayesian regression: 1.2824

mean-squared error

Least-squares: 3.1028

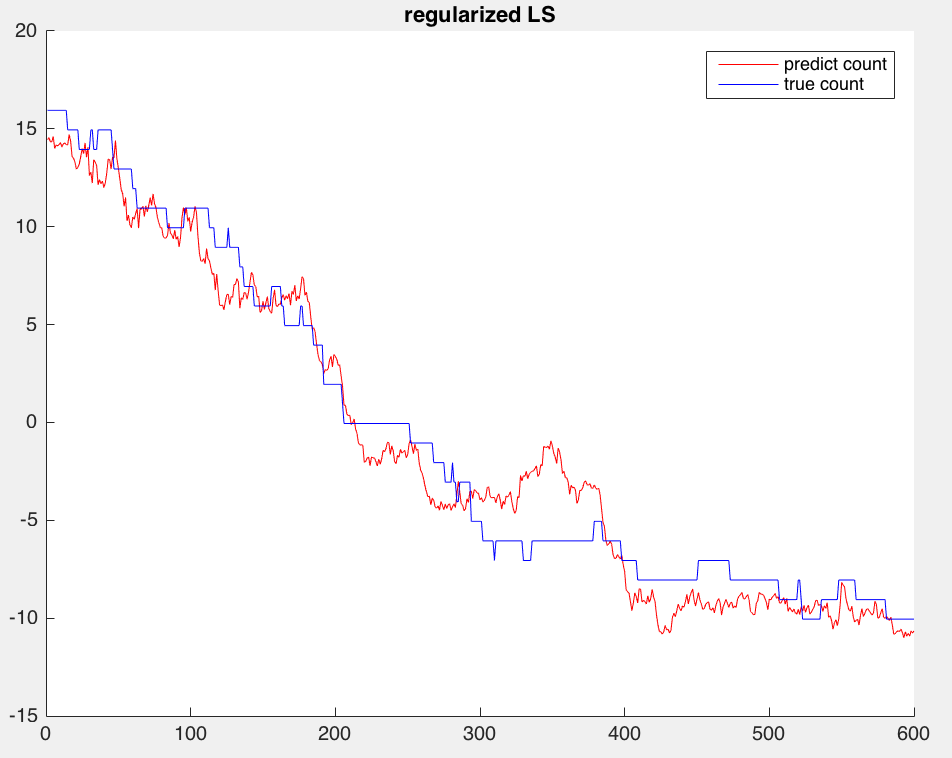
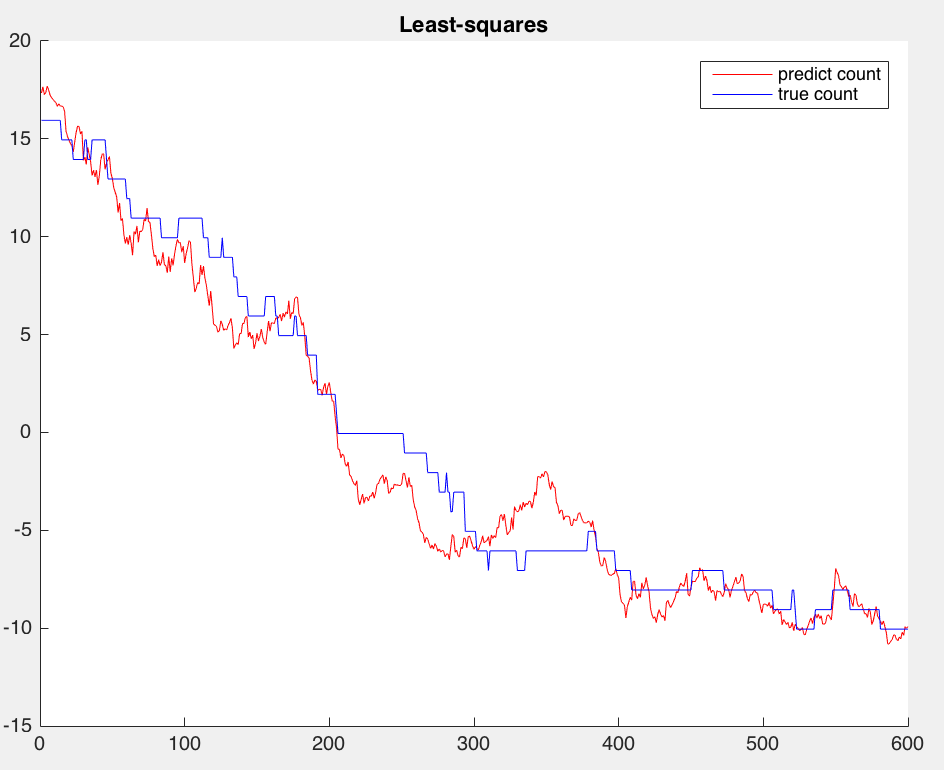
Regularized LS: 3.1463

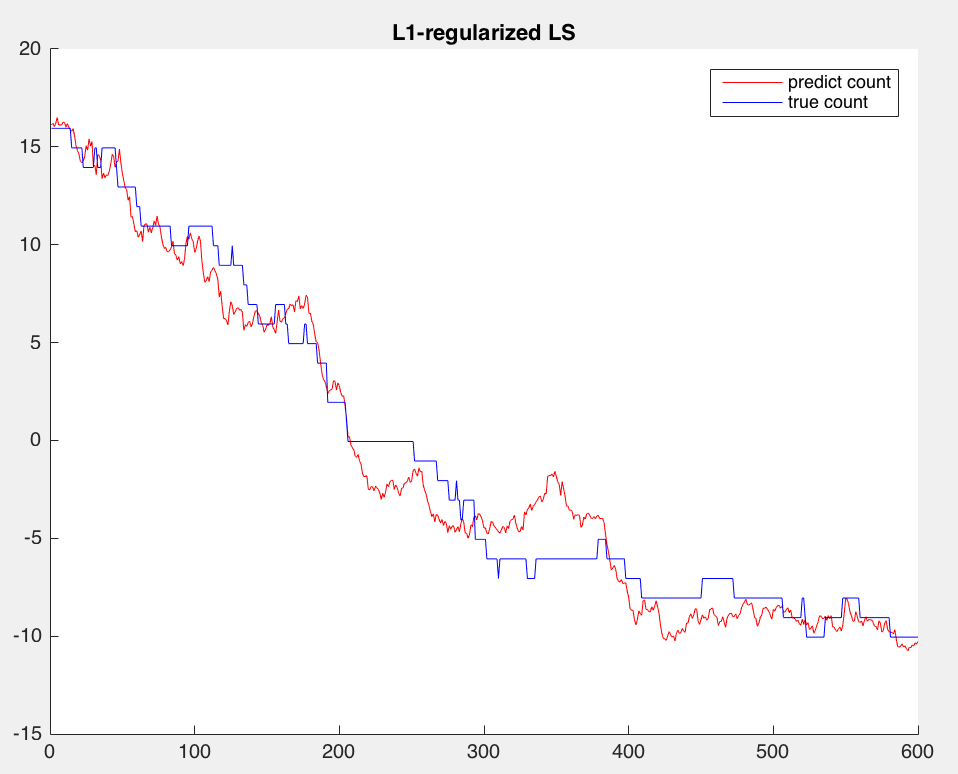
L1-regularized LS: 2.4645

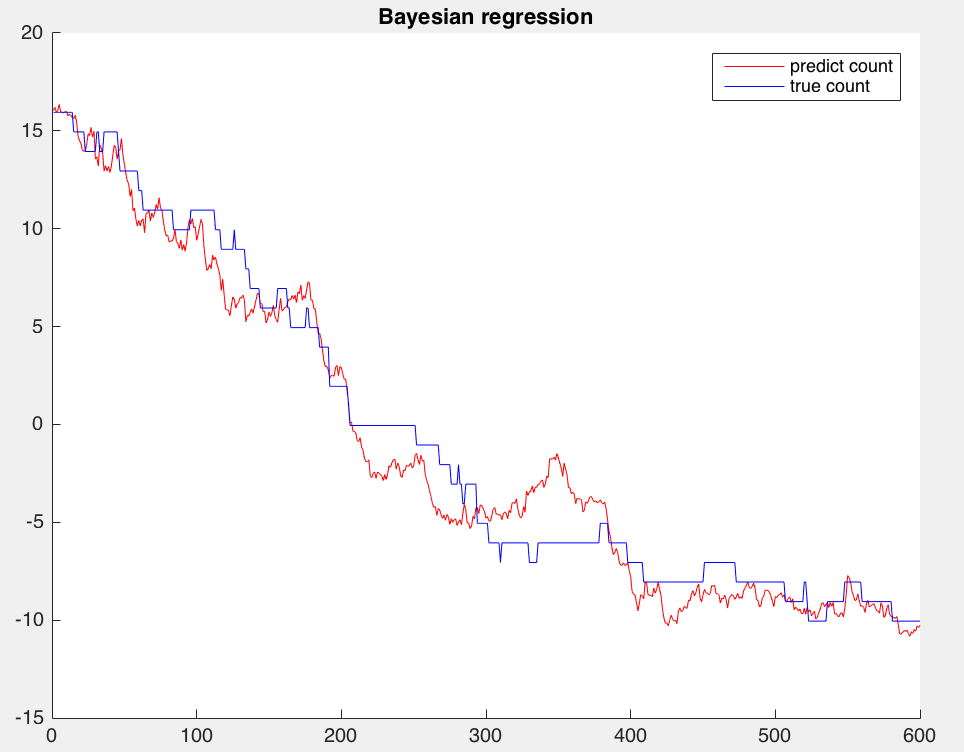
robust regression: 3.5812

Bayesian regression: 2.6187

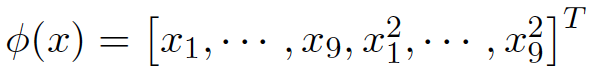
From the result above, I find that L1-regularized LS method works the best.







(b)

For 

mean-absolute error and mean-squared error

