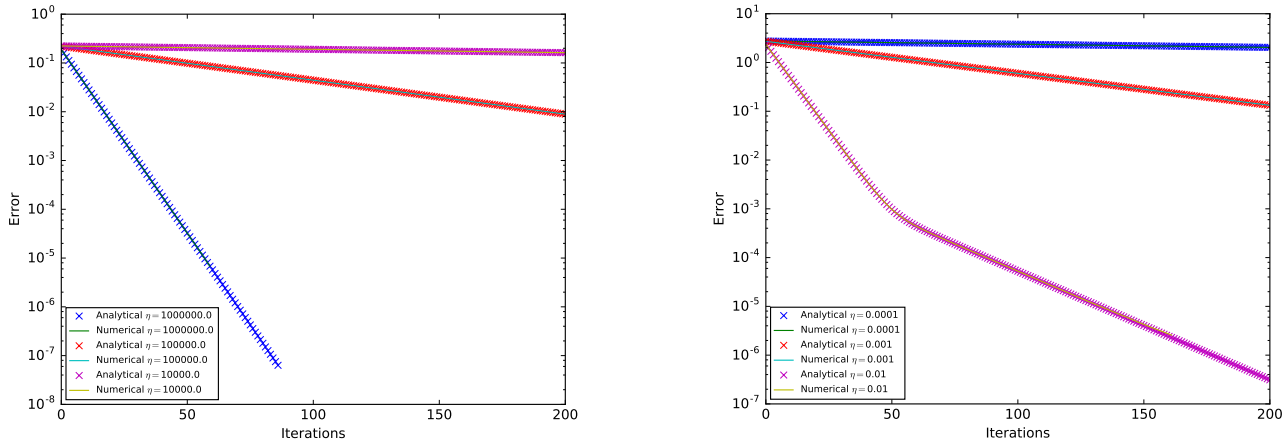
# 6.867: HW1

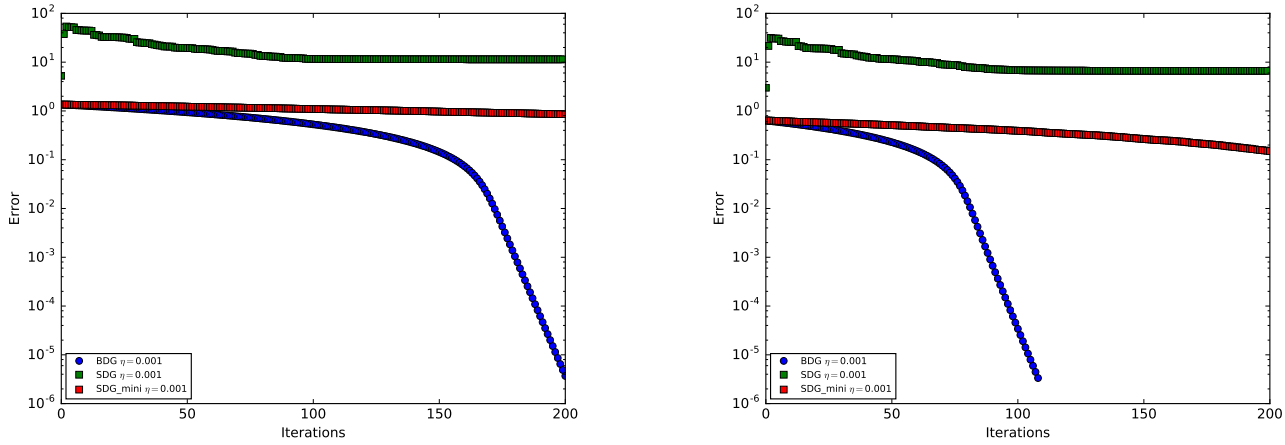
This report covers some techniques used to solve convex minimization problems. To begin, two functions, the negative Gaussian and the quadratic bowl, where the analytic derivative can be obtained were used to validate the implementation of gradient descent. Figure 1 shows the effect of different learning rates on the speed of error reduction, where larger learning rate leads to faster error reduction. Here error is defined $\frac{\|x-x_{exact}\|}{\|x_{exact}\|}$. This error metric is subsequently used for the remainder of the work, where $x_{exact}$ is replaced with $\theta_{ML}$ and $ML$ being the parameters obtained from Maximum-Likelihood. While not shown, the numerical differentiation is implemented using central difference thus the error scales as $O(\delta^2)$, where $\delta$ is the finite step size.




(a) Negative Gaussian, $x_{exact} = [10, 10]$

(b) Quadratic bow, $x_{exact} = [26.\overline{66}, 26.\overline{66}]$

FIG. 1: Effect of learning rate.



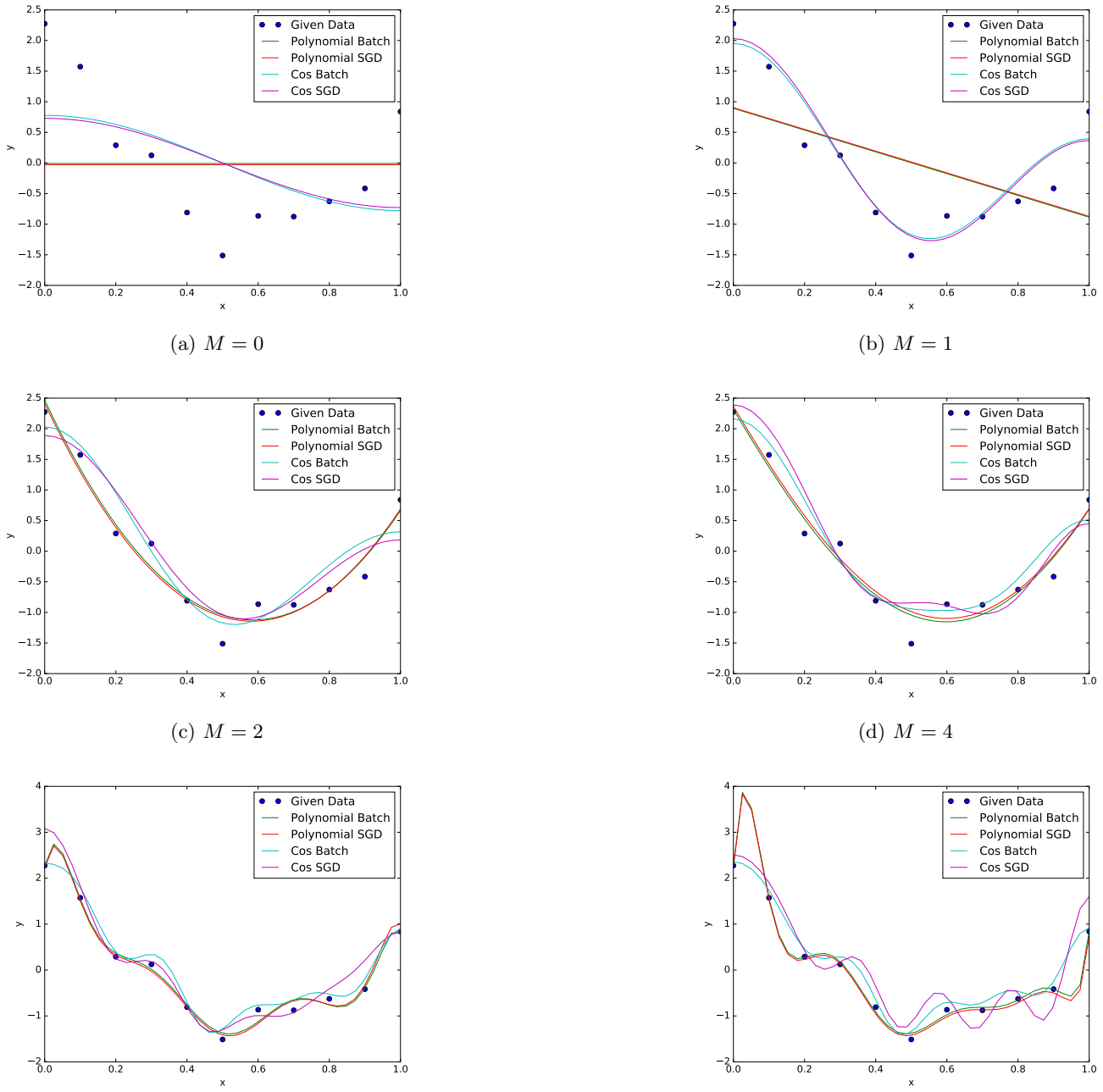

(a) Initial guesses sampled from $N(0, 1)$

(b) Initial guesses sampled from $N(0, 5)$

FIG. 2: Effect of initial guesses.

With the numerical differentiation and gradient descent validated, the next task was the perform batch gradient descent (BDG), stochastic gradient descent (SDG) and stochastic mini-batch gradient descent (SDG-mini) on the least square error problem for the given set of data. The effect of different initial guesses is shown in Figure 2, which is small while small enough learning rate. The learning rate in (SDG) was setting using the relation $\eta_{t+1} = \frac{\eta_0}{t}$. SDG is somewhat more temperamental than BDG or SDG-mini when it comes to selecting an initial learning rate; too large and the error diverges while too small and the error does not converge to a tolerable level. In the end, after manually

trying different initial learning rates, $\eta_0 = 0.001$ was determined to be suitable. Note that the SDG-mini batch (using randomly drawn batches of 10) smoothes out the initial behavior of SDG where the error does not monotonically decrease. In the future, other methods for choosing learning rate, such as Backtracking line search, can be used to improve the DG behavior.




(a) $M = 0$ (b) $M = 1$

(c) $M = 2$ (d) $M = 4$

(e) $M = 8$ (f) $M = 9$

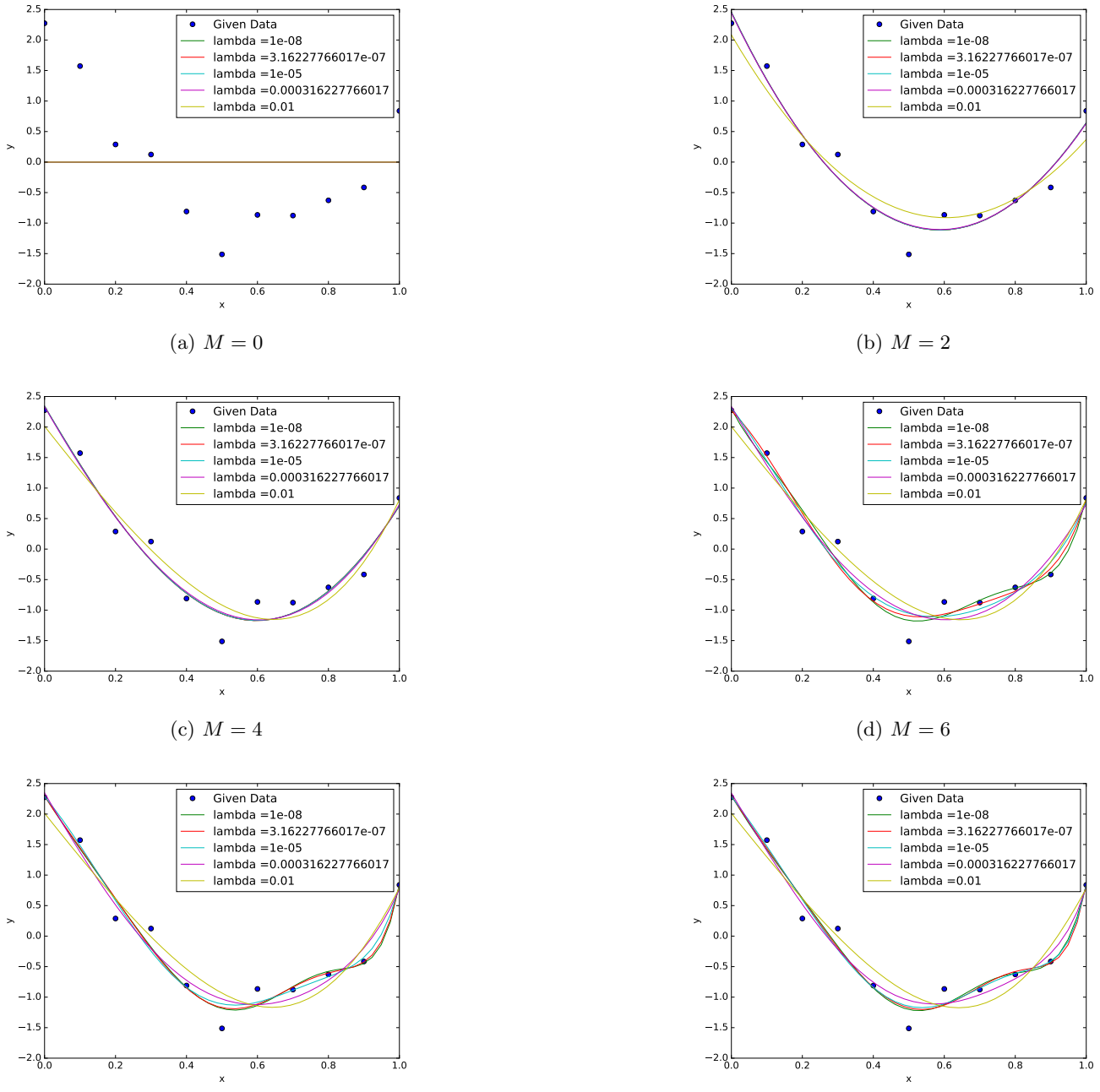
FIG. 3: Polynomial and Cosine basis for different orders $M$.

We now move to effect of basis choice upon regression. Figure 3 shows the obtained functions for different orders of polynomial and cosine bases using the implemented BDG and SDG to minimize the sum of square errors (SSE). While the SSE decreases with increasing order (see Table II, at the end of report.), the consequence is overfitting, as seen in Figures 3e and 3f. Without any further data, an order of $M = 2$ is suitable, given its simplicity, to recover the exhibited trend. Table **??** shows the weights for the cosine expansion for $M = 8$ obtained through Maximum-

Likelihood, Ridge-regression with hyperparameter $\lambda = 0.1$ and LASSO with $\lambda = 0.1$. While, RR and ML obtain larger weights for the first two terms, LASSO is capable of recovering the sparsity of the expansion by setting four of the terms to zero. The effect of different choices of $\lambda$ in RR for a range of polynomial orders is shown in Figure 4, where larger $\lambda$ pulls the function away from the ML result.

TABLE I: Table of weights for the cosine expansion for $M = 8$.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| ML | 0.76885361 | 1.08747411 | 0.09929377 | 0.14328126 | -0.05071775 | 0.361557 | 0.01225909 | 0.01510985 |
| RR | 0.75556579 | 1.08625726 | 0.09913457 | 0.160578 | -0.04793554 | 0.37457382 | 0.01380646 | 0.03491975 |
| LASSO | 0.59565947 | 0.96592302 | 0. | 0.02172924 | 0. | 0.24000549 | 0. | 0. |

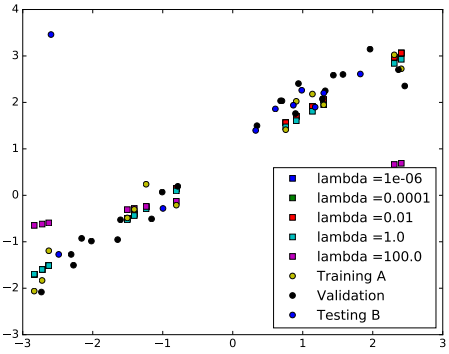



(a) $M = 0$ (b) $M = 2$

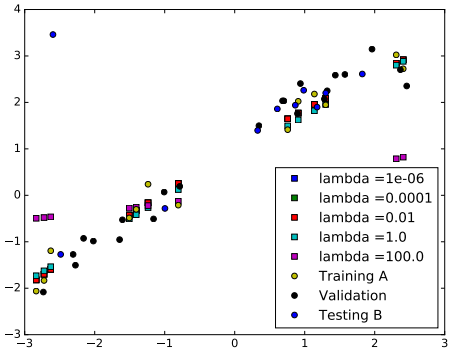
(c) $M = 4$ (d) $M = 6$

(e) $M = 8$ (f) $M = 10$

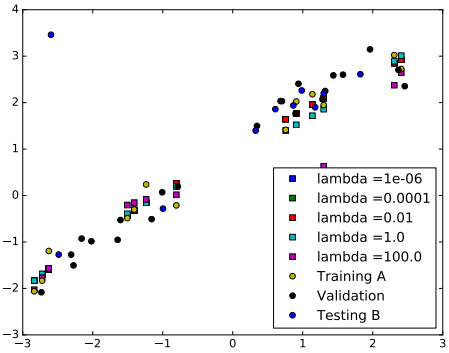
FIG. 4: Ridge regressions for different $\lambda$ and $M$.



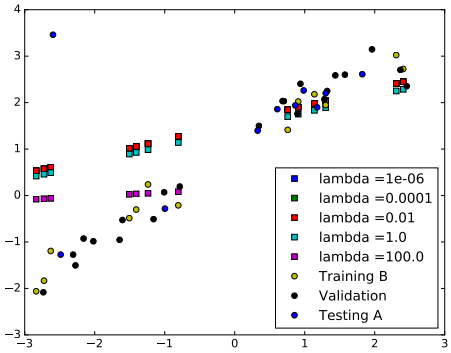

(a) Training A, $M = 1$



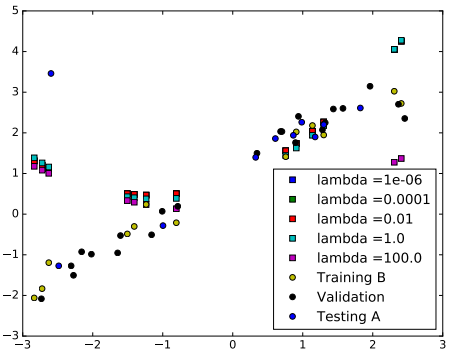

(b) Training A, $M = 2$



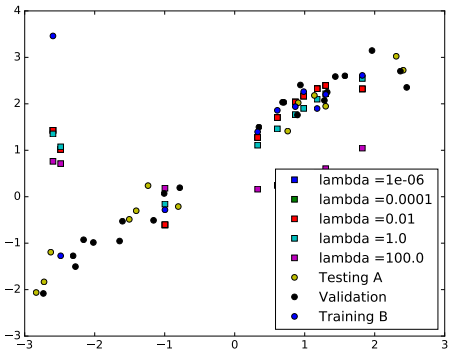

(c) Training A, $M = 3$




(d) Training B, $M = 1$



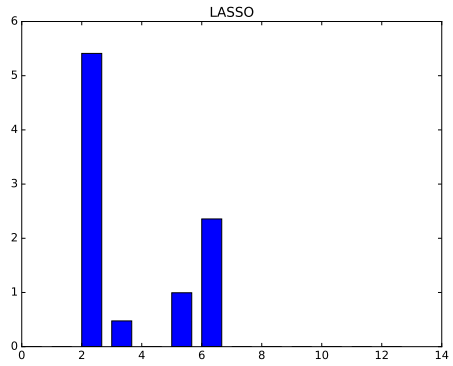

(e) Training B, $M = 2$



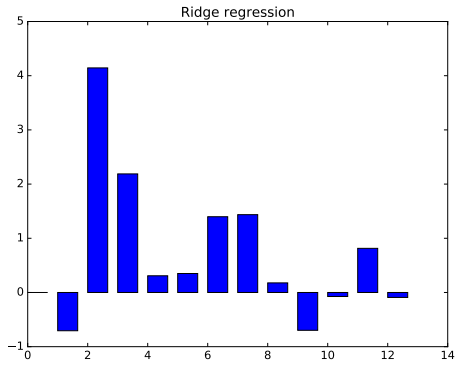

(f) Training B, $M = 3$

FIG. 5: RR for different $\lambda$ and $M$, against Training A and Training B.

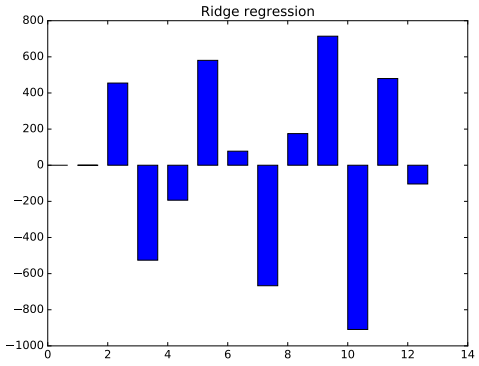
The results shown Figure 5 exemplify the importance of dividing data sets into a training, validation and testing set while selecting a model. If we were only provided dataset B, we would likely have selected a higher order polynomial for the basis of our model because of the yet unknown outlier near [-2.5,3.5], but with validation and testing, we see that a simple linear, $M = 1$ with a sufficiently small $\lambda < 1.0$, suffices to capture the trend.



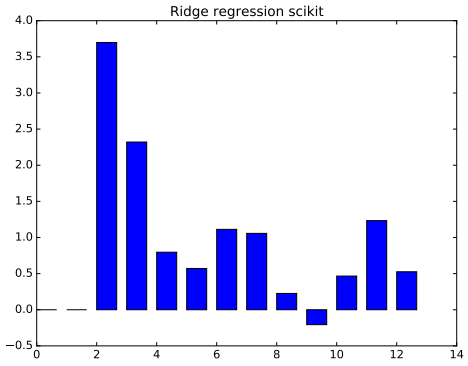

(a) Weights for LASSO, $\lambda = 0.1$.



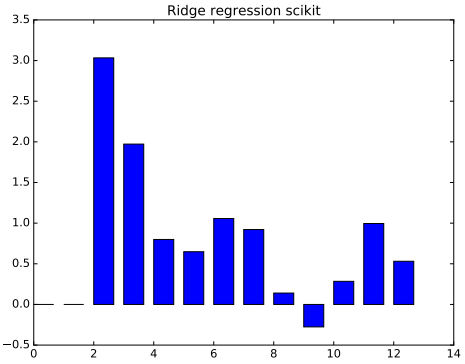

(b) Weights for RR, $\lambda = 1e-5$.



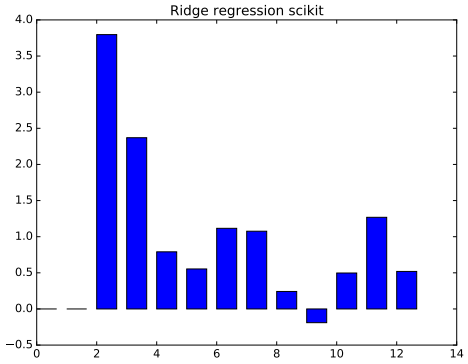

(c) Weights for RR, $\lambda = 1e-8$.




(d) Weights for RR, $\lambda = 0.1$.




(e) Weights for RR, $\lambda = 1$.




(f) Weights for RR, $\lambda = 1e-8$.

FIG. 6: Weights obtained from RR and LASSO.

Finally, we investigate a situate where no closed form solution exists (i.e. no functional form for the ML estimates). In Figure 6, the weights obtained by using RR and LASSO with a sinusoidal expansion on the given data (shown in Figure 7). Generally, LASSO is able to nearly recover the true weights (only 4 terms are non-zero), while RR assigns value to a broad set of weights. It is interesting to note that the home-brew implementation of RR does not produce the same weights as those obtained from RR using the `sklearn.linear model`, further emphasizing the importance of trying different versions of the same method or algorithm while generating a model.

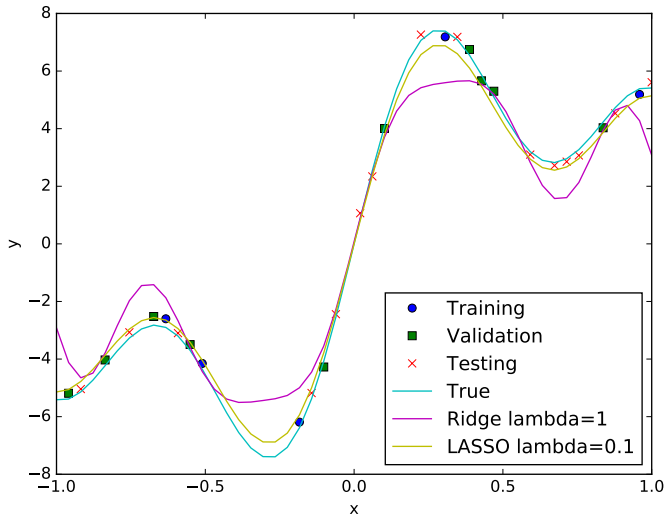



FIG. 7: Obtained functions for training data.

| $M =$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Polynomial | 13.47696 | 9.86198301 | 0.69286344 | 0.65484575 | 0.64936234 | 0.5299862 | 0.42185923 | 0.39279451 |
| Cosine | 9.8359897 | 1.56447217 | 1.51314829 | 1.28655369 | 1.27272969 | 0.51791369 | 0.51706833 | 0.5157841 |

TABLE II: Table of SSE for Polynomial and Cosine basis for results in Figure 3 .