6.867 Homework 1

Anonymous authors

September 28, 2017

1. Implement Gradient Descent

1.1. Basic gradient descent

We investigate the use of the basic gradient descent method for finding the optimum of the negative Gaussian function (Equation 1), and the quadratic bowl function (Equation 2). For the remainder of this paper, optimum will refer to the global minimum of the function unless specified otherwise.

$$f(x) = -\frac{10^4}{\sqrt{((2\pi)^n|\sum|)}}\exp\left(-\frac{1}{2}(x-u)^T\Sigma^{-1}(x-u)\right)$$

Equation 1. Negative Gaussian function

$$f(x) = \frac{1}{2}x^TAx - x^Tb$$

Equation 2. Quadratic bowl function

The general basic gradient descent algorithm is as follows (note subscript k indicates iteration number):

1) Choose an initial starting point $x_k = x_0$
2) Calculate $\nabla f(x_k)$
3) Set $x_{k+1} = x_k + \alpha_k d_k^T \nabla f(x_k) x_k$, where α_k is the step size and d_k is a direction
4) Repeat until some convergence criteria is met, usually $\|\nabla f(x)\|_2 \leq \epsilon$, or $\|x_{k+1} - x_k\|_2 \leq \epsilon$, where ϵ is some chosen tolerance.

Note that the basic gradient descent algorithm uses the gradient of the function directly and thus can be subject to numerical issues if the gradient is very large. In some cases, it may be beneficial to scale the gradient, but this is not currently considered. We currently set $d_k = -1$, and α_k to a constant value. This will be referred to as steepest gradient descent (SteepGD) with a constant step size. We are interested in the behavior of SteepGD as the choice of x_0, α_k, and convergence criteria change. We'll change each of these values independently of the others and see how the true error $\|x_k - x^*\|_2$(where $x^\star$ is the true optimum solved analytically for the negative Gaussian and quadratic bowl functions), behaves with each iteration. We start with the negative Gaussian function and changing the step-size; the initial point is set to $\begin{bmatrix}0\\0\end{bmatrix}$, with a stopping criteria of $\|x_{k+1} - x_k\| \leq 0.0001$. Step-sizes of 1, 0.5, 0.1, and 0.01 are used. Values higher than this resulted in almost immediate divergence. The resulting real error, $\|x_k - x^*\|$, is plotted against iteration number in Figure 1 on the left. It was observed that step-sizes of 1 and 0.5 converged in accordance with the stopping criteria, but not to the correct optimum, while step-sizes of 0.1 and 0.01 converged to the correct optimum. The 0.01 step-size required almost an order of magnitude more steps to reach the optimum compared with the 0.1 step-size. These results suggest that too large of a step-size can result in divergence as the SteepGD method overshoots the optimum with its estimates, and that the step-size must be small enough to converge, but not too small as to take many iterations converge. It is likely that an optimum step-size could be chosen at each step to maximize reduction in the objective function.

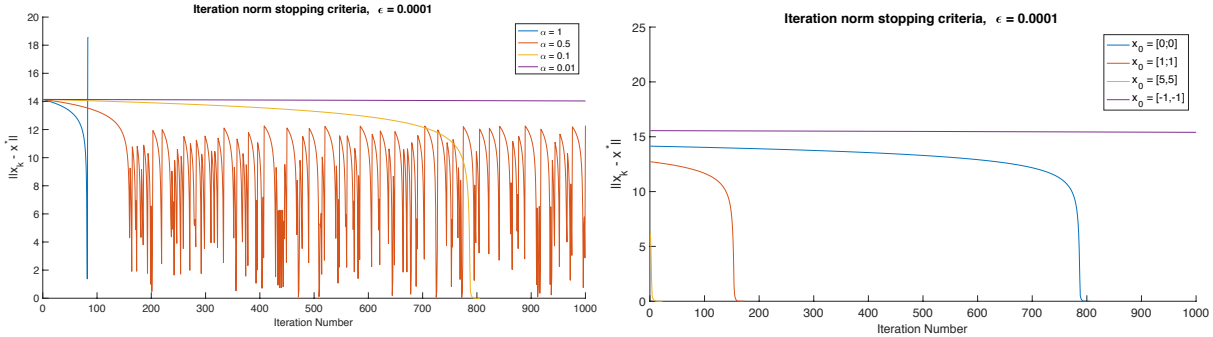

Figure 1. Real error plotted against iteration number for changes in step-size (left) and starting points (right) for SteepGD on the negative Gaussian function

Next, we change the initial starting point with a constant step-size of 0.1, and $\epsilon = 0.0001$. Starting points chosen were $\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}5\\5\end{bmatrix}, \begin{bmatrix}-1\\-1\end{bmatrix}$ with the error vs. iteration number shown in Figure 1. It was observed that as the starting point has a direct effect on number of iterations required for convergence, with closer starting points resulting in fewer iterations for convergence. This makes sense intuitively, as a longer distance should require more steps.

We move on to the quadratic bowl function and change the step-size and starting point in the same way as before. This time, $\alpha = 1, \alpha = 0.5$ result in divergence, with $\alpha = 0.1, 0.05, and\ 0.01$ result in convergence to the correct optimum. Out of these, $\alpha = 0.05$ resulted in the fewest iterations for convergence, again suggesting that an optimal step-size exists. For the quadratic bowl function, it seems that the starting point does not affect the convergence rate much, with all the starting points resulting in similar convergence characteristics. This suggests that the gradient is proportional to distance from the optimum and so the starting point has little effect on convergence rate. For the effect of convergence criterion, in general a small ϵ results in more iterations until convergence is reached.

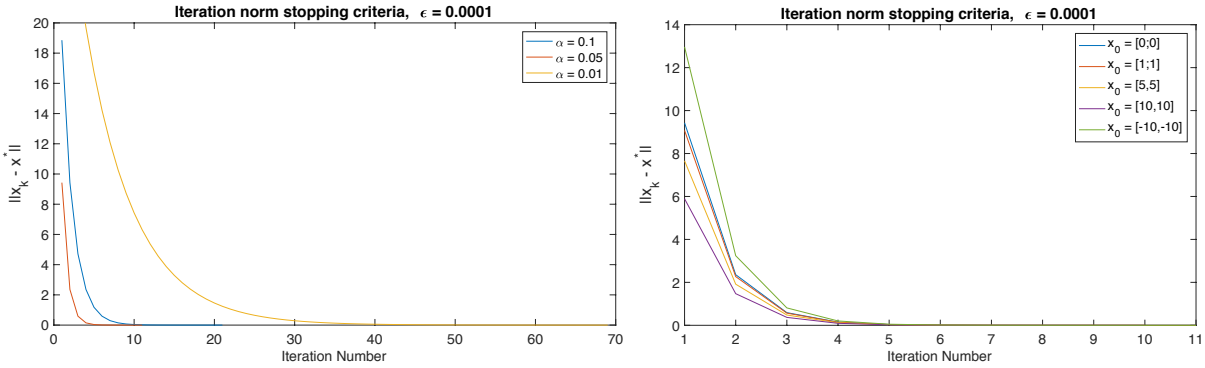

Figure 2. Real error plotted against iteration number for changes in step-size (left) and starting points (right) for SteepGD on the quadratic bowl function

1.2. Finite difference approximation of the gradient

When the gradient function is not available in closed form or easily derived, a numerical approximation can be used. A common approximation is the central difference approximation (CDA) shown in Equation 3. This approximation is derived from the Taylor series expansion around $f(x)$ and the error of the approximation is of $O(\delta^2)$. To verify the use of the CDA, the mean error between the CDA and analytical gradient ($\frac{1}{N}\sum\|\nabla f(x_i) - \nabla f_{CD}(x_i)\|$) across 100 random points between -5 and 5 (for each component of x) is plotted against various values of δ in Figure 3. The mean error is shown to decrease as expected with $O(\delta^2)$. Additionally, the error is very small, showing that the central difference approximation is a viable alternative to the analytical gradient if δ is small enough.

$$\nabla f(x) \approx \frac{f(x+\delta) - f(x-\delta)}{2\delta} = \nabla f_{CD}(x;\delta)$$

Equation 3. Central difference approximation for $f(x)$

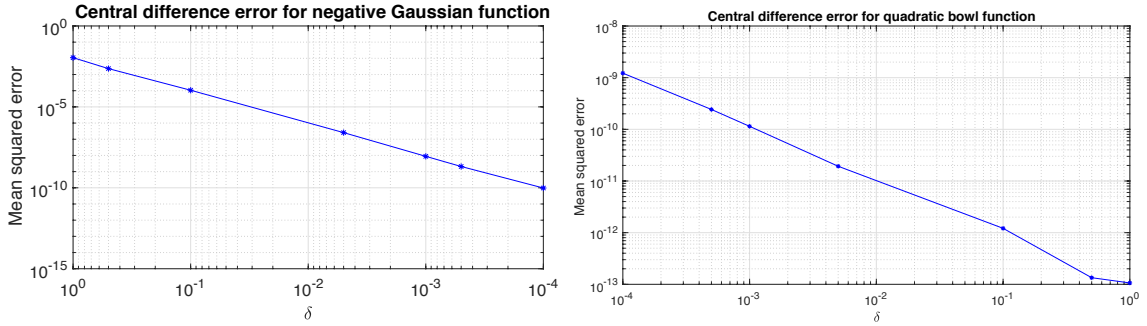

Figure 3. Mean squared error of central difference approximation for negative Gaussian function (left) and quadratic bowl (right)

1.3. Batch vs. Stochastic gradient descent

We are now interested in the question of calculating linear regression coefficients θ, that best quantify the relationship between some n data sample pairs (x,y), where x is a vector of length m and y is a scalar. That is finding $\theta^* = argmin_\theta(J(\theta;X,y) = \|X\theta - y\|^2)$, where X is a $n\ x\ m$ matrix and y is a vector of corresponding y values. Note that $\nabla J(\theta;X,y) = \sum_{i=1}^{n} -2(y_i - x_i\theta)x_i^T$, which is a sum over all n data points. In many situations, n is very large and evaluating the gradient is expensive. To address this issue, stochastic gradient descent (SGD) can be used, which is similar to BGD, except instead of the entire gradient, only the gradient of one data point x is used in each iteration. The general algorithm for SGD is as follows:

1. Select initial point θ_0 and random data point pair (x_{SGD}, y_{SGD})
2. Calculate the point-wise gradient $\nabla J(\theta_k; x_{SGD}, y_{SGD})$

3. Update theta $\theta_{k+1} = \theta_k - \eta_k \nabla J(\theta_k; x_{SGD}, y_{SGD})$, where $\eta_k = (\tau_0 + k)^{-\kappa}$, satisfying the Robbins-Monro conditions. η_t is known as the learning rate.
4. Repeat until some convergence criterion is met.

We are interested in how SGD compares to BGD in terms of point-wise gradient evaluations and total error. To compare, we consider 100 (x,y) data point pairs, with x being 10 x 1 and calculate θ^* with both BGD and SGD with a convergence criterion of $\|\theta_k - \theta^*\| \leq 10^{-6}$, where $\theta^* = (X^T X)^{-1} X^T y$ is the analytical optimum of $J(\theta; X, y)$. For BGD, we use a constant step of 10^{-5} and for SGD we use $\tau_0 = 30000, \kappa = 0.9$. The error $\|\theta_k - \theta^*\|$ is plotted against the number of point-wise gradient evaluations for BGD and SGD in Figure 4. We observed that BGD met the convergence criterion in 4600 point-wise gradient evaluations while SGD did not converge even after 30000 iterations, with the error remaining relatively constant after ~120 point-wise gradient evaluations. This is postulated by the authors be due to the stochastic nature of SGD or to the rapidly decreasing learning rate required to avoid divergence. The authors have not rigorously investigated the root cause of the "frozen" error.

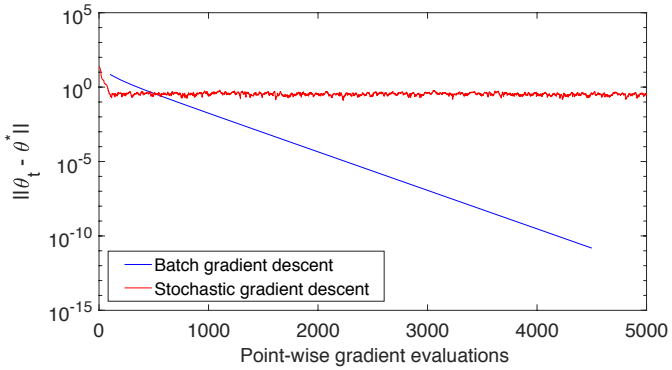

Figure 4. BGD vs. SGD Total Error as a function of point-wise gradient evaluations

2. Linear Basis Function Regression

2.1. Polynomial basis regression

We now investigate regression models that are a linear combination of basis functions. Models of this class take in a set of (x,y) pairs, transform x by some basis function $\phi(x)$ and find weights θ such that θ minimizes $\|Y - \boldsymbol{\phi}\theta\|_2^2$. The maximum likelihood solution is $\theta = (\boldsymbol{\phi}^T \boldsymbol{\phi})^{-1} \boldsymbol{\phi}^T Y$. A synthetic data set is used, generated from the distribution: $y(x) = \cos(\pi x) + \cos(2\pi x) + \epsilon$, where ϵ is Gaussian noise; 11 data points are generated.

We start by not assuming the basis functions are known and attempt a polynomial basis of order M, defined as $\phi_i(x) = x^i, i = 0, 1, \ldots, M$. M values of 0, 1, 3, and 10 are used. The resulting regression function and generated data points are shown in Figure 5. As the value of M increases, the fit is improved at the cost of overfitting.

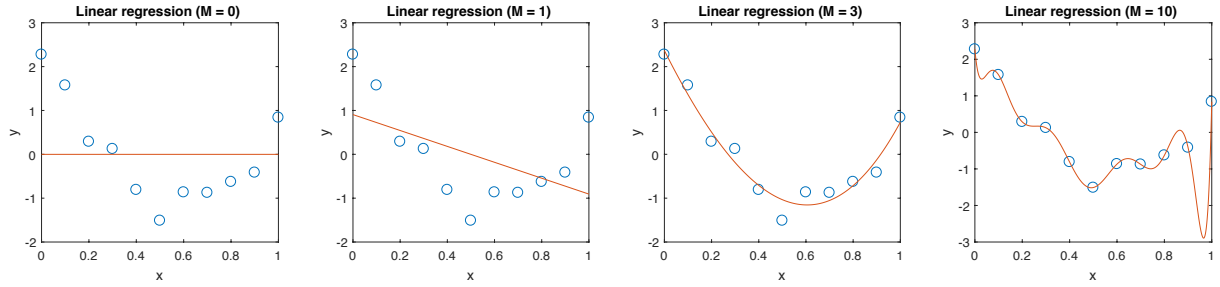

Figure 5. Regression with polynomial basis, M = 0, 1, 3, 10

2.2. Sum of squares error, central difference approximation error

For these four values of M, the sum of squares error (SSE) and error between the analytical gradient and central difference approximation of the gradient are calculated. The results are shown in Table 1. The central difference approximation performs very well for low values of M and gets worse as M increases.

	M = 0	M = 1	M = 3	M = 10
SSE	13.477	9.862	0.655	4.0481×10^{-5}
$\|\nabla SSE - \nabla_{CDA} SSE\|$	8.88×10^{-16}	8.88×10^{-16}	6.255×10^{-13}	1.670×10^{-9}

Table 1. SSE of polynomial basis regression and error between CDA and analytical gradients

2.3. Batch vs. Stochastic gradient descent convergence characteristics

We next investigate the use of BGD and SGD for solving the linear regression problem with the polynomial basis for various values of M. Even though the inputs are transformed nonlinearly, the objective function is still quadratic

and the full gradient is a sum of weighted contributions from each point. For this objective function, the starting point, step sizes and convergence thresholds can greatly affect the convergence characteristics. If the largest and smallest Eigenvalues of the Hessian of the objective function have a large difference between them, it can be shown that the upper bound on iterations required for convergence is large as well. This is because the Eigenvalues represent how "stretched out" the quadratic objective function is and thus how sensitive the convergence characteristics are to starting points and step-sizes. We start by using BGD and SGD for M values of 0, 1, 3, and 4 as in the previous section and investigate the number of iterations required for convergence for various values of step-size (for BGD, Robbins-Monro constants for SGD), starting point, and convergence threshold. For this investigation, a convergence criterion of true error, $\|\theta_{GD} - \theta^*\| \leq \epsilon$ is used. The iterations required for convergence for various parameter sets are shown in Figure 6. The main trends/takeaways are as follows:

- For small values of ϵ, BGD converges in fewer point-wise gradient iterations than SGD. This is likely due to the stochastic nature of SGD; it is very possible that the structure of the function is such that near the optimum, the point-wise gradient is very small, leading to only small movements in θ.
- The number of point-wise gradient iterations required for convergence for BGD is very sensitive to changes in constant step-size, with iterations required affected directly by step-size. SGD is less sensitive to changes in τ_0, as the effect of τ_0 on η is very small as $k \gg \tau_0$.
- As the initial guess θ_0 is closer to the optimum θ^*, the fewer number of iterations required for convergence for both SGD and BGD.
- In general, as M increases, the number of iterations required for convergence increase for both BGD and SGD and the step-size required for convergence decreases for BGD as M increases. This is because as M increases, the magnitude of the gradient increases because the magnitude of the weights is very high (symptom of overfitting) and the gradient is directly proportional to the magnitude of the weights. The high gradients require smaller step-sizes to avoid divergence. If the Hessian eigenvalues are found, it is very likely that as M increases, the difference in magnitude between the lowest and highest eigenvalues increases. It can be shown that the constant step-size required for convergence of BGD is dictated by this difference.

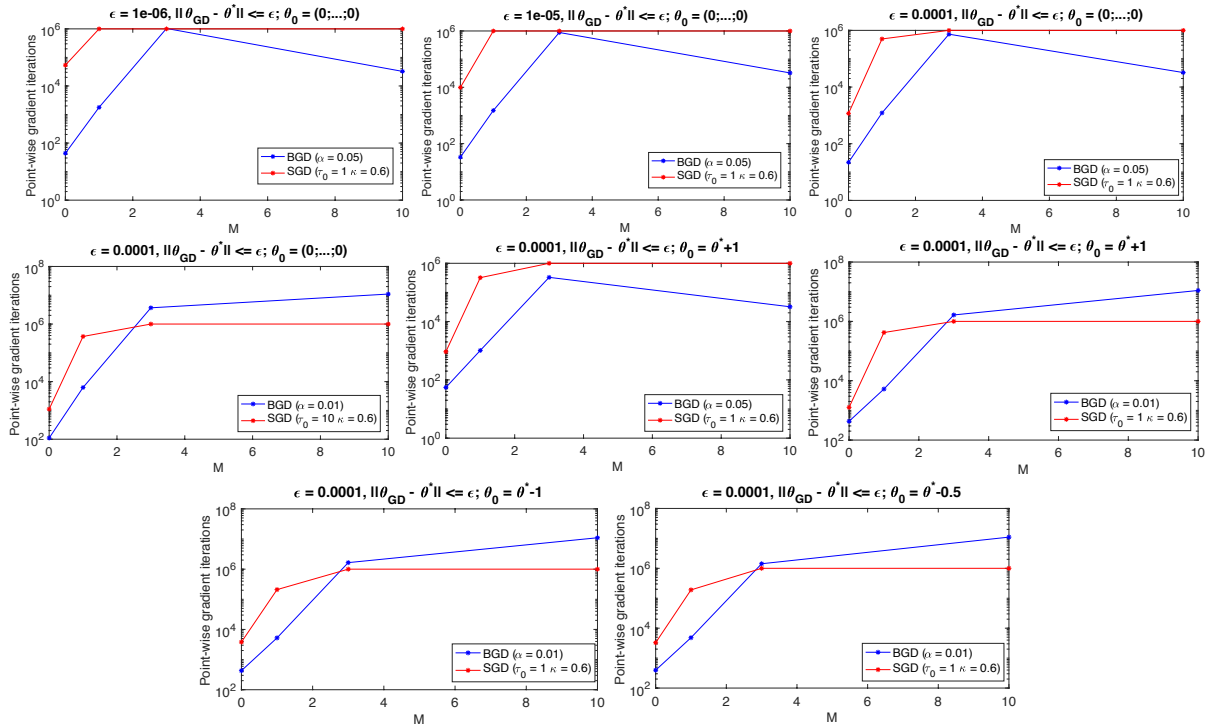

Figure 6. Point-wise gradient iterations required for convergence for various values of M, step-size (α and τ_0), convergence criterion (ϵ) and initial point (θ_0)

2.4. Cosine basis regression

Now, instead of a polynomial basis, we investigate the use of a cosine basis, which is the same basis used to generate the data points. We perform the maximum likelihood weight vector calculation for the cosine basis (Equation 4) for $M = 8$, and we include $\phi_0(x) = 1$. Considering the function used to generate the data, we would expect that $\theta_i = 1$ for $i = 1, 2$ and $\theta_i = 0$ otherwise as this would replicate the generating function exactly (not considering the noise). The regression calculated from the maximum likelihood (ML) weight vector is shown in

Figure 7. The ML weight vector is: $\theta^* = [-0.153; 0.768; 1.104; 0.099; 0.160; -0.050; 0.378; 0.012; 0.032]$. While the magnitude of θ_1 and θ_2 are the largest among the weights, they are not the only non-zero entries as would be expected. This is because the function used to generate the data includes noise which makes the data deviate from the base generating function and the maximum likelihood approach finds the weights that best fit the data, noise included.

$$\phi_i(x) = \cos(i * \pi x), i = 1 \ldots, M$$

Equation 4. Cosine basis function definition

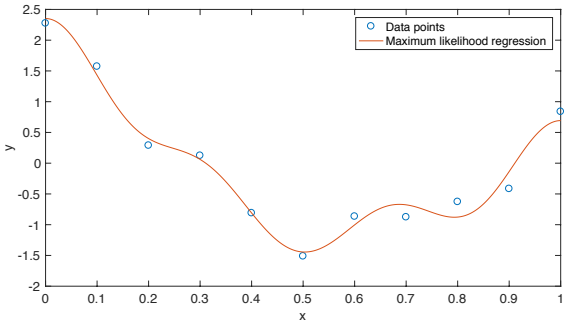

Figure 7. ML regression for cosine basis

3. Ridge Regression

3.1. Ridge regression and effect of regularization parameter λ

We now investigate the regularized least squares regression problem, where the objective function now has an additional term that increases as the regression coefficient magnitudes increase. This new objective function is shown in Equation 5. The regularizer used in this case is the L_2 norm of the weight vector θ, and regression that minimizes this error function is known as ridge regression; the closed form solution is $\theta^* = (\lambda \boldsymbol{I} + \boldsymbol{\phi^T \phi})^{-1} \boldsymbol{\phi^T} y$. We calculate the regularized weight vector for a variety of $M$and λ for the data in section 2 with a polynomial basis. The resulting regressions along with the mean squared error (MSE) are shown in Figure 8. In general, as the order of the regression M increases, the MSE decreases and as λ increases, the MSE increases. Additionally, as M increases, the regression shows greater oscillations while as λ increases, the oscillations decrease, so they show opposite effects.

$$E(\theta; \phi(x), y, \lambda) = \frac{1}{2}\sum_{i=1}^{N}[y_i - \theta^T \phi(x_i)]^2 + \frac{\lambda}{2}\theta^T\theta$$

Equation 5. L_2 norm regularized sum of squares error function

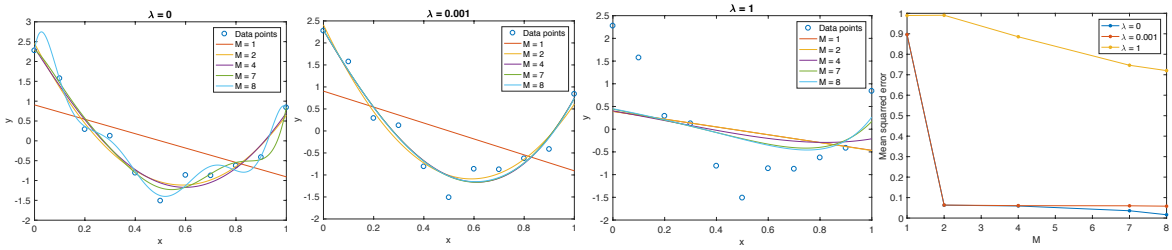

Figure 8. Ridge regularized regression on section 2 data and MSE for various values of M and λ

3.2. Ridge regression model selection

We now investigate the model selection problem with the same model as before and with three sets of data, A, B, and validation. The problem of interest is choosing model parameters M and λ that result in a good fit to the data and robustness to outliers. To begin, we use data set A as the training set and data set B as the test set. We pick a training set to fit model weights to for a variety of M and λ and choose the set that performs best in terms of MSE on the validation set (M = 2, $\lambda = 0.1$). We then see how this model along with other low MSE models perform on the test set. This process is illustrated in Figure 9 (note, only $\lambda = 0$ regressions are shown for the training and validation data but not the only ones used). The MSE of the chosen model was found to be 2.575 while the MSE of the $M = 7, \lambda = 0.1$ and $M = 8, \lambda = 0.1$ are 2.326 and 2.338. While the chosen model performs best on the validation set, it does not perform best on the test data. In a real situation, we would not know this apriori and we can only choose based on information we have. We then repeat the process, now using B as the training set and A as the test set shown in Figure 10. In this case, we choose $M = 3, \lambda = 1$ as the best model, which has an MSE of 2.87 on the test

data. By comparison, the MSE values of $(M = 4, \lambda = 1)$, $(M = 3, \lambda = 8)$, $(M = 7, \lambda = 0.1)$ are 5.43, 2.57, and 64.54. There are several takeaways from this investigation: 1) ridge regression (RR) weights can be highly influenced by outliers in training data, even with a high λ value, 2) RR model selection based on MSE works well for training data free of outliers but not well when outliers are present, 3) RR has a tendency to ignore outliers in test data when trained on "well-behaved" training data and 4) the regularizer has the effect of controlling oscillations in the polynomial basis regression, as expected as it reduces the weight on higher order terms.

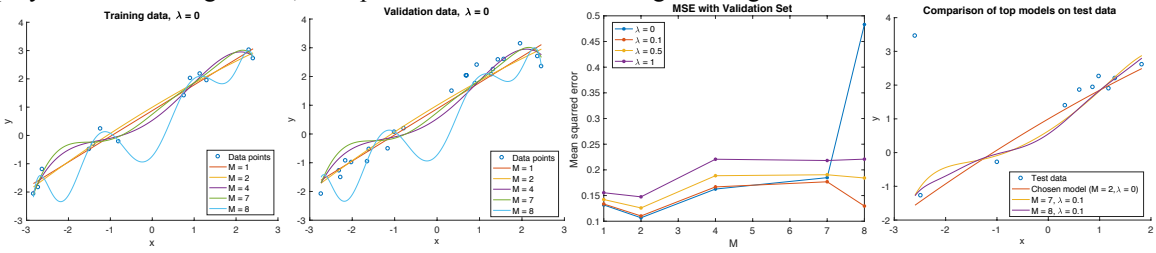

Figure 9. Model selection with training data A and test data B, chosen minimum MSE model $M = 2$, $\lambda = 0.01$

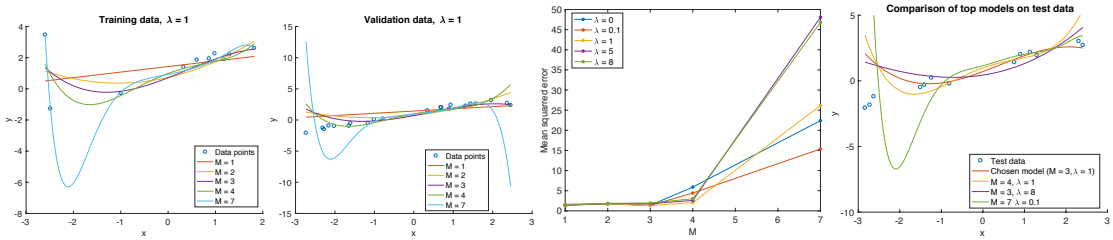

Figure 10.Model selection with training data B and test data A, chosen minimum MSE model $M = 3$, $\lambda = 1$

4. Sparsity and LASSO

In the previous section we investigated the use of the L_2 norm in the regularization term of the objective function. We now investigate the use of the L_1 norm instead (Equation 7). This type of regression is known as LASSO. We use a new set of data, again split into training data, validation data, and test data. In this section we consider a sine basis, shown in Equation 6 Ridge and LASSO regression for various values of λ are used and compared with the true weights used to generate the data; they are trained on the training data. The results are shown in Figure 11. We observe that the weight vector generated by LASSO achieves significant sparsity with even a small value of λ in this case, while ridge does not, only reducing the magnitudes of all the weights as λ is increased. In fact, for $\lambda = 0.05$, LASSO weights almost mirror the true weights. Looking at the regressions compared to the data, they all fit well and in this case we would likely choose LASSO as the model due the sparsity, with no loss in accuracy on the validation data compared to the ridge models.

$$\phi(x) = (x, \sin(0.4\pi x * 1), \sin(0.4\pi x * 2), \ldots, \sin(0.4\pi x * 12)) \in \mathbb{R}^{13}$$

Equation 6. Sine basis function used in section 4.

$$E(\theta; \phi(x), y, \lambda) = \frac{1}{2}\sum_{i=1}^{N}[y_i - \theta^T\phi(x_i)]^2 + \lambda\sum_{j=1}^{M}|\theta_j|$$

Equation 7. L_1 norm regularized sum of squares error objective function

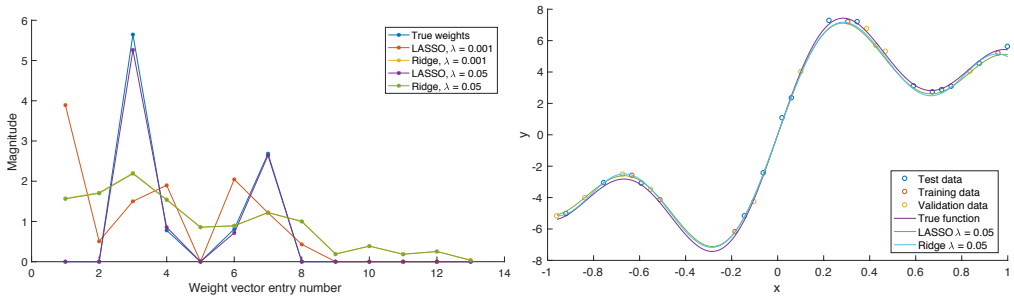

Figure 11. LASSO vs. Ridge regression for various values of λ