6.876 Homework 1

Anonymous Homo Sapiens[1]
[1] *University of Rock and Roall*
(Dated: September 27, 2016)

In this homework, we explore the function regression with different regularizations. We also implement the gradient decent algorithm for the minimization problem.

I. IMPLEMENT GRADIENT DECENT

Gradient decent (GD) is a method to obtain the local minimum of functions. Consider a function $f(\mathbf{x})$ from $\mathcal{R}^n \to \mathcal{R}$, the gradient decent starts with an initial guess $\mathbf{x}_0$. Then in k^{th} step with the guess $\mathbf{x}_{k-1}$, it obtains a new guess by descending certain step size $\delta_k > 0$ along the gradient at point $\mathbf{x}_{k-1}$

$$\mathbf{x}_k = \mathbf{x}_{k-1} - \delta \nabla f(\mathbf{x}_{k-1})/||\nabla f(\mathbf{x}_{k-1})||, k \geq 1, \quad (1)$$

where $||\nabla f(\mathbf{x}_{k-1})||$ is the norm of the gradient. It terminates when the norm of the gradient is smaller than $\epsilon \ll 1$ compared with the function value, i.e.

$$||\nabla f(\mathbf{x}_k)||/|f(\mathbf{x}_k)| < \epsilon. \quad (2)$$

Note that if the function value is close to zero, we can use the absolute value to determine convergence.

Conventional GD requires the explicit analytical formula of the function being optimized over and the calculation of its gradient. Variations of GD often use approximations of the gradient to save time.

A. Part1

1. Initial guess

Since GD is a local search algorithm, it will be trapped at local minimum. Consequently, unless there is only a single global minimum it is important to be able to provide good guess as a initial start. When there some prior knowledge about the solution, one can make some intelligent guess; often, uniform random guesses are used when there is no prior knowledge about the solution.

2. Step size

We use fix step size of $\delta_k = \delta$. The step size will influence the running time to reach condition Eqn. 2. We plot the runtime and the step size's relation in Fig. 1. In fact, if the step size is too large, in my experiments it may fail to terminate since the step size is discrete. The choice of step size may rely on different problems, and without the global knowledge it is in general not easy to choose a good step size.

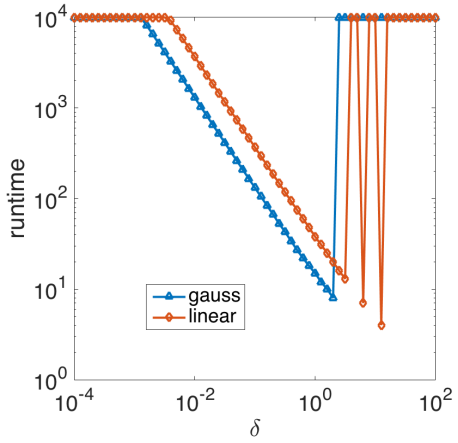


Figure 1. $\epsilon = 10^{-3}$, runtime vs. δ. We terminate the program if runtime $> 10^4$ steps. When δ is too small, it fails to terminate because the $\mathbf{x}_k$ change too slowly; when δ is too big, it fails to terminate because Eqn. 2 cannot be satisfied with the discrete points.

3. Convergence criterion

Large ϵ will lead to short runtime and early termination far away from the true minimum; small ϵ will lead to long runtime with high precision. We plot the final error, quantified by the error by the norm of the deviation from the true minimum $||\mathbf{x}_k - \mathbf{x}_m||$, and ϵ's relation in Fig. 2. We fix $\delta = 10^{-3}$. To see the precision change, we choose the runtime cutoff to be 5×10^4.

4. Evolution of the norm of the gradient

We see in both case the norm of the gradient decrease super exponentially with the step size (Fig. 3).

B. Part2

The gradient can be obtained by the limit of

$$\nabla f(\mathbf{x}) = \lim_{\delta \to 0} \sum_i \mathbf{e}_i \frac{f(\mathbf{x} + \delta \mathbf{e}_i) - f(\mathbf{x})}{\delta}, \quad (3)$$

where $\mathbf{e}_i$ is the unit vector for each coordinate. As δ decreases, it becomes more and more accurate.

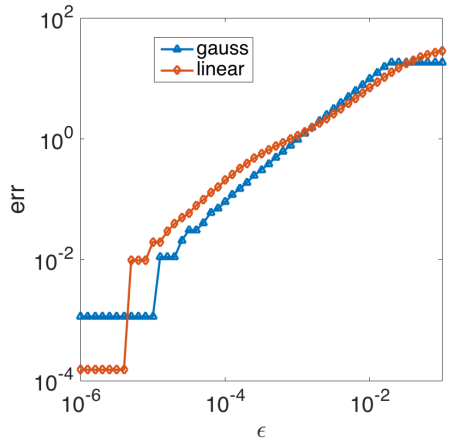


Figure 2. $\delta = 10^{-2}$, error vs. ϵ. The error stops decreasing at certain ϵ, since the step size is too large at these ϵ.

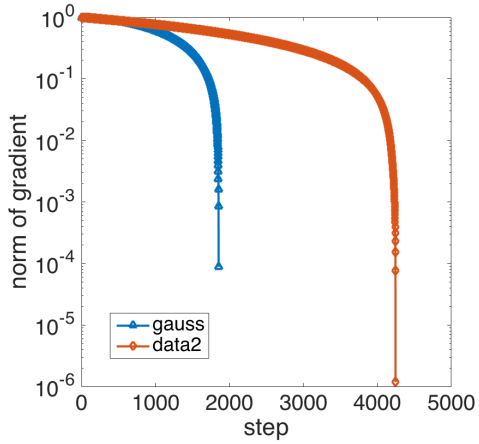


Figure 3. $\delta = 10^{-2}$, $\epsilon = 2.5 \times 10^{-6}$.

C. Part3

For batch gradient, from the definition of $J(\theta)$, we can obtain the gradient

$$\nabla J(\theta) = \sum_i 2(x^{(i)T}\theta - y^{(i)})x^{(i)}. \tag{4}$$

We change the fixed step size δ and ϵ to obtain a good solution in reasonable time. After some attempts, the variable are chosen to be $\delta = 2.5 \times 10^{-3}, \epsilon = 10^{-1.8}$ and the evolution of $J(\theta)$ and norm of gradient are in Fig. 4.

For the SGD method, we need to derive the point-wise gradient function.

$$\nabla_\theta J(\theta_t; x^{(i)}, y^{(i)}) = 2(x^{(i)T}\theta_t - y^{(i)})x^{(i)}. \tag{5}$$

Then the update of each step is given by

$$\theta_{t+1} = \theta_t - \eta_t \frac{\nabla_\theta J(\theta_t; x^{(i)}, y^{(i)})}{||\nabla_\theta J(\theta_t; x^{(i)}, y^{(i)})||}, \tag{6}$$

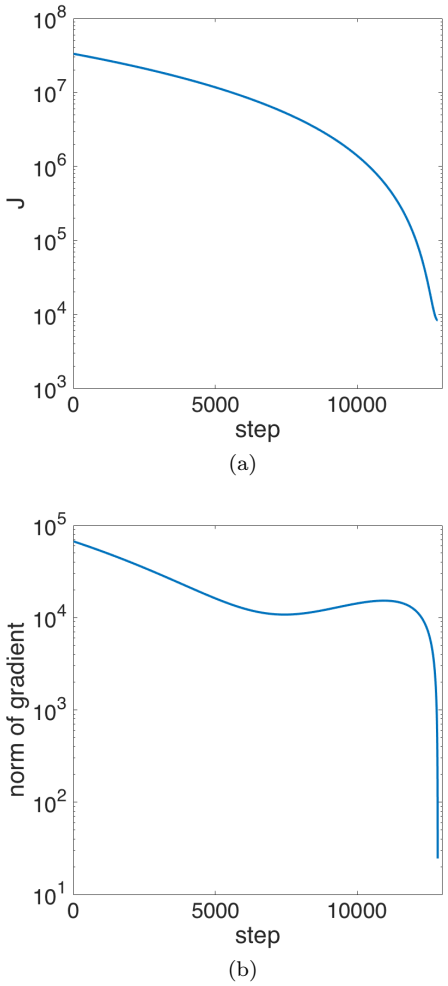


Figure 4. (a)J and step, batch (b) the norm of gradient and step, batch. The minimum J obtained is around 8333.4, also it doesn't get smaller if you continue the evolution due to the finite step size.

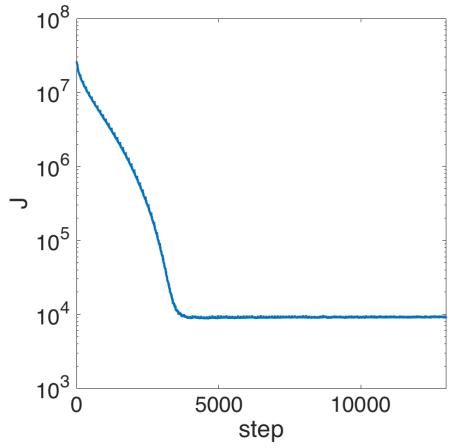
where we chose the learning rate $\eta_t = (\tau_0 + t)^{-\kappa}$. The results are given in Fig. 5. Note that here we also normalize the gradient to enable easy control of step size.

Comparing two methods, we find that the batch method converge slower than the SGD method, while achieving slightly better minimum values of the J function. In each iteration, one need to evaluate the point-wise gradients n times in the batch method, and only one time in the SGD method.

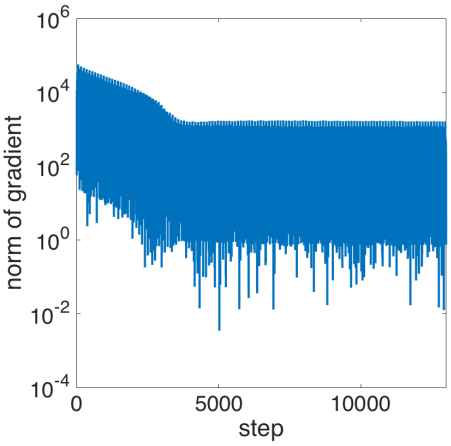
II. LINEAR BASIS FUNCTION REGRESSION

A. Part 1, polynomial bases

In order to use the maximum likelihood method, we need to make some assumption about the noise statistics. As usual, we assume the noise at each point is



(a)



(b)

Figure 5. $\tau_0 = 0.1, \kappa = 1/2 + 1/100$. (a)J and step, SGD (b) the norm of gradient and step, SGD. The minimum J obtained is around 8579, also it doesn't get smaller if you continue the evolution due to the finite step size.

i.i.d. Gaussian, thus for vector $w = \{w_1, \cdots, w_M\}$, the probability distribution of $Y = \{y_1, \cdots, y_n\}$ for the $X = \{x_1, \cdots, x_n\}$ is

$$P(Y|X) = \prod_{k=1}^{n} \mathcal{N}\left(\sum_{i=1}^{M} w_i \phi_i(x_k), \beta^{-1}\right), \tag{7}$$

thus the maximum likelihood weight is

$$w_{\text{ml}} = \arg\min_w \sum_{k=1}^{n} (y_k - \sum_{i=1}^{M} w_i \phi_i(x_k))^2. \tag{8}$$

Take the derivative of w_j we have

$$\sum_{k=1}^{n} y_k \phi_j(x_k) - \sum_{k=1}^{n} \sum_{i=1}^{M} w_i \phi_i(x_k) \phi_j(x_k) = 0, \forall 1 \le j \le M, \tag{9}$$

If we define matrix $\Phi_{ij} = \sum_{k=1}^{n} \phi_i(x_k)\phi_j(x_k)$ and the vector $v_j = \sum_{k=1}^{n} y_k \phi_j(x_k)$, we have the equation

$$w^T \Phi = v. \tag{10}$$

Then w can be solved by the above linear system of equations. the results are in Fig. 6.

B. Part 2. SSE

The SSE is

$$\begin{aligned} E_M(w; X, Y) &= \sum_{k=1}^{n} (y_k - \sum_{i=1}^{M} w_i \phi_i(x_k))^2 \\ &= 2\left(\frac{1}{2} w^T \Phi w - w^T v\right) + \text{constant}, \end{aligned} \tag{11}$$

where matrix Φ and vector v is defined in Sec. II A. This is exact the same thing being minimized over in Sec. II A. The gradient is given by

$$\nabla_w E_M(w; X, Y) = 2(w^T \Phi - v). \tag{12}$$

We have verified that this is the correct gradient numerically.

C. SSE with batch and SGD

The function being minimized over is a second order polynomial in w and has only one global minimum, thus the convergence is guaranteed. In fact, it is the same form as the linear example in Sec. I. Thus we only need to replace the matrix and vector and the program works. All properties and discussions about the initial guesses, step sizes and convergence thresholds are the similar.

D. Part 4. cosine bases

We only need to change to the cosine bases and calculate the matrix $\Phi_{ij} = \sum_{k=1}^{n} \phi_i(x_k)\phi_j(x_k)$ and the vector $v_j = \sum_{k=1}^{n} y_k \phi_j(x_k)$ and solve w from

$$w^T \Phi = v. \tag{13}$$

We see in Fig. 7 that when $M = 2$ it is the most accurate.

III. RIDGE REGRESSION

A. Part 1

The Ridge regression basically add a penalty term $\lambda||w||^2$ to the SSE in Eqn. 11

$$E_M(w; X, Y) = 2\left(\frac{1}{2} w^T \Phi w - w^T v\right) + \lambda||w||^2, \tag{14}$$

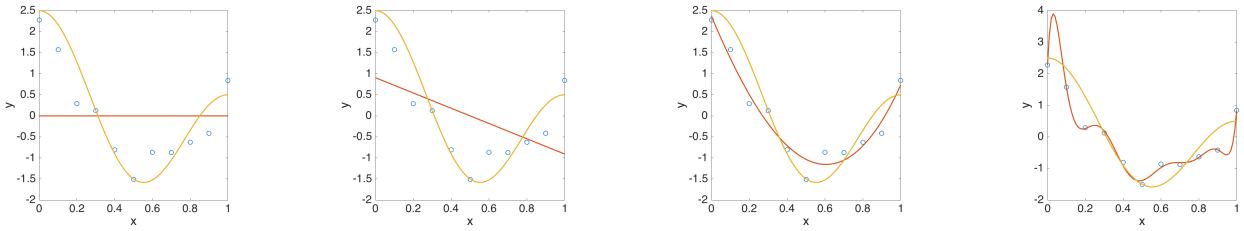


Figure 6. Reproducing Fig.1.

1	0.7790
2	[0.7790;1.1741]
3	[0.7634;1.1741;0.0938]
4	[0.7634;1.1413;0.0938;0.1971]
5	[0.7704;1.1413;0.1008;0.1971;−0.0492]
6	[0.7704;1.0894;0.1008;0.1452;−0.0492;0.3634]
7	[0.7689;1.0894;0.0993;0.1452;−0.0507;0.3634;0.0123]
8	[0.7689;1.0875;0.0993;0.1433;−0.0507;0.3616;0.0123;0.0151]
9	[0.7911;1.0875;0.1216;0.1433;−0.0284;0.3616;0.0345;0.0151;−0.2006]
10	[0.7911;1.0812;0.1216;0.1370;−0.0284;0.3552;0.0345;0.0088;−0.2006;0.0568]

Figure 7. Table of weights w for $1 \leq M \leq 10$.

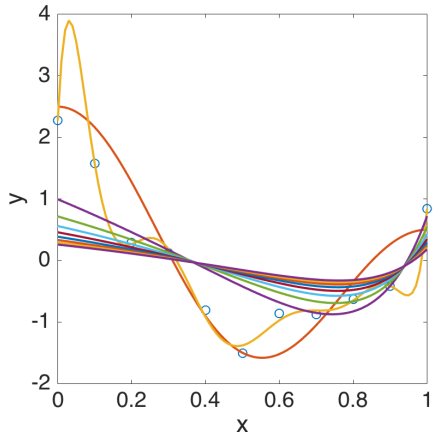


Figure 8. As λ increases from 0 to 2 with interval of 0.25, the $M = 10$ fitting of problem 3 becomes more and more towards a straight line.

where we have left out the constants. The gradient equals zero gives the equation

$$w^T (\Phi + \lambda I) = v. \tag{15}$$

The numerical experiment shows that for all M, larger λ prevents the fitting to be rippled, i.e. tends to make the fitting more straight, we give an example of $M = 10$ in Fig. 8, we see that as λ increases from 0 to 2 with interval of 0.25, the $M = 10$ fitting of problem 3 becomes more and more towards a straight line.

B. Part 2

We calculate the SSE without the penalty as an evaluation of the performance of a fit, characterized by M, λ and the learning data set X_0, Y_0 it is obtained, on the training data set X, Y,

$$E\left(X, Y; M, \lambda, X_0, Y_0\right) \tag{16}$$

$$= \sum_{k=1}^{n} \left(y_k - \sum_{i=1}^{M} w_i \left(M, \lambda, X_0, Y_0\right) \phi_i \left(x_k\right)\right)^2 . \tag{17}$$

1. *data set A and data set B*

If we use data set A as the test set for model selection in data set B, $M = 2, \lambda = 5.7$ is optimum. If we use data set B as the test set for model section in data set A, $M = 10, \alpha = 0.1$ is optimum. This is because in data set B there is a point isolated from all the other points (see Fig. 11)

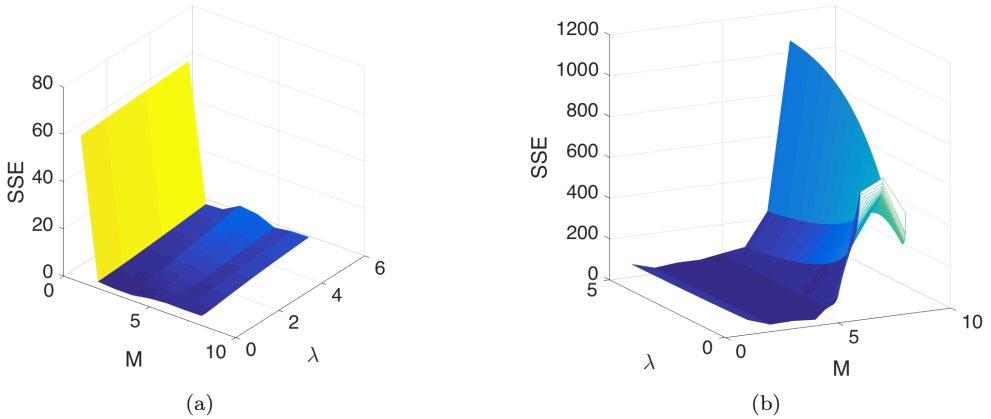
C. validate data set

We use the validate set for model selection for the fitting in data set A and data set B. The SSE for different choice of (M, λ) are given in Fig. 9. We find that for training set A, the optimal choice is $\lambda = 0, M = 4$, with SSE 2.34 on the validate set; for training set B, the optimal choice is $\lambda = 0.56, M = 3$, with SSE 28.9 on the validate set. The fitting with the optimum model selection on the validate data set are plot in Fig. 10. Due to the "error" point in data set B, its fitting is much worse than data set A. We also plot the fitting of the original training data set with the optimum model selection on the validate set in Fig. 11.

IV. SPARSITY AND LASSO

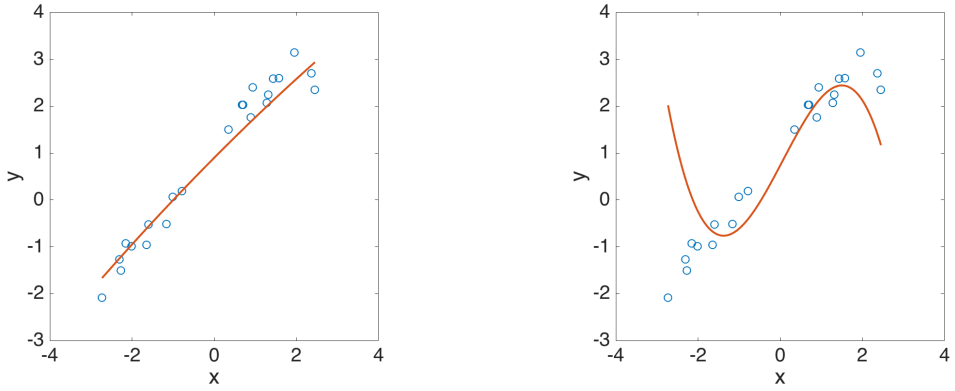
The LASSO regularization uses the absolute value and has an objective function as follows

$$\frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - w^T \phi(x^{(i)}))^2 + \lambda \sum_{j=1}^{M} |w_j|. \tag{18}$$



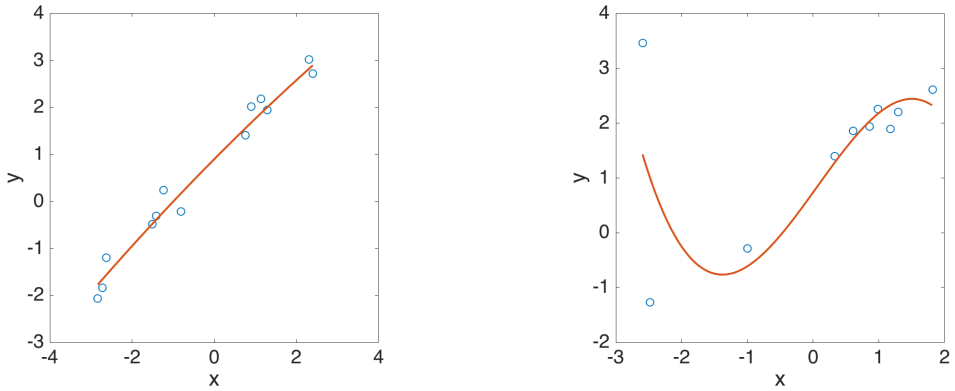
(a) (b)

Figure 9. λ chosen from 0 to 2 with interval 0.01; M chosen from 1 to 10. (a)Training set A. Optimal choice $\lambda = 0, M = 4$, SSE equals 2.34. (b) Training set B. Optimal choice is $\lambda = 0.56, M = 3$, SSE equals 28.9.



(a) (b)

Figure 10. Optimum fitting of the validate data set. (a)Training set A and $\lambda = 0, M = 4$. (b) Training set B and $\lambda = 0.56, M = 3$.



(a) (b)

Figure 11. Optimum model fitting from the validate data on the original data set. (a)Training set A and $\lambda = 0, M = 4$. (b) Training set B and $\lambda = 0.56, M = 3$.

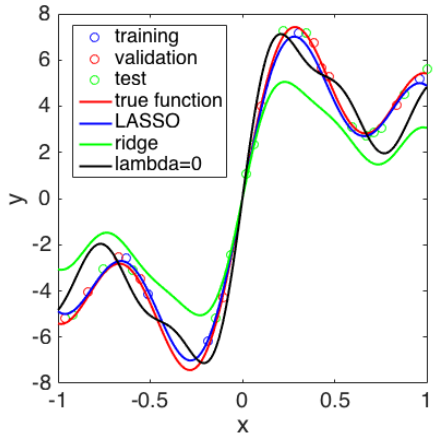


Figure 12. LASSO with $\lambda = 0.1$, ridge with $\lambda = 1$.

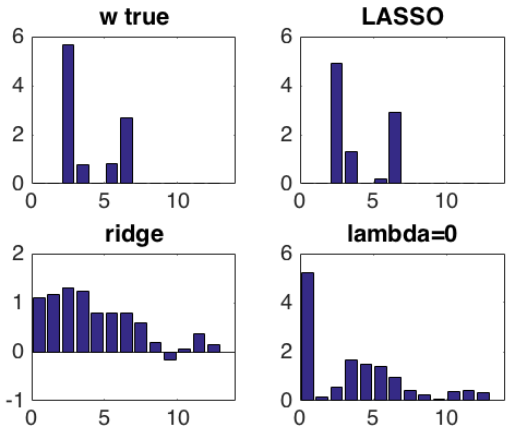


Figure 13. Comparison of different strategies and the true value.

One minimizes the above function to obtain the weight w. We use matlab function to perform the LASSO regression and repeat the results shown in the homework in Fig. 12 and Fig. 13. Several λ values have been tried and the values are selected when the fitting is good.