IN4320 Machine Learning Exercise 1 Report

Exercise Regularizations & Sparsity

Lu Liu (4621832)

February 28, 2017

1. Some Optima & Some Geometry

1.1. Question 1

The loss function L used in this assignment is:

$$L(m_-, m_+) := \sum_{i}^{N} ||x_i - m_{y_i}||^2 + \lambda||m_- - m_+||_1$$

with λ the regularization parameter, N the number of samples in the training set, and $||\cdot||_1$ the L_1-norm.

1.1.1. a

In this part, the loss function in one dimension is drawn as a function of m_+ for for $\lambda \in \{0, 2, 4, 6\}$.

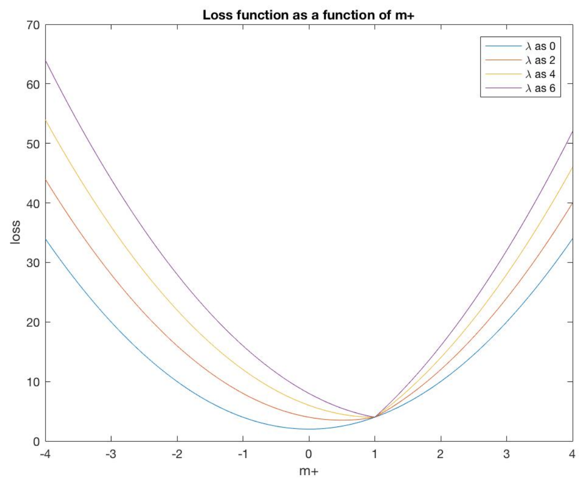


Figure 1: Loss Function as a function of m_+ for for $\lambda \in \{0, 2, 4, 6\}$

1.1.2. b

In this part, for every of the four functions, the minimizer and their minimum values are derived as well as the point where the derivative equals 0. Here I use MATLAB to find the points. Since with different λ the MATLAB code for drawing figure and finding the point with derivative as 0 is the same, here only the code for λ as 0 is shown in Appendix A.

And the answer I obtained is:

Table 1: Results for Question 1(b)

λ	minimizer	minimum values	the point with derivative as 0	its value
0	0	2	0.0100	2.002
2	0.5	3.5	0.5100	3.5002
4	1	4	1	4
6	1	4	1	4

When searching for the point with derivative as 0, I try to find the point with the minimum absolute value of derivative. Thus the points with derivative as 0.02 are found here, which is almost the same point.

1.2. Question 2

1.2.1. a

In this loss function, the norm is 1. Thus the shape of $\lambda||m_- - m_+||_1$ is shown in Figure 2. It is the square placed on the origin. In Figure 2, the ellipses mean the contour for the first part of loss function L, which is $\sum_i^N ||x_i - m_{y_i}||^2$. The intersection of them is the minimum point expected. If λ is very small, there might be no intersection. Thus the mean values obtained at that time is not the optimized. With the increase of λ, the radius of the square increases. The intersection appears and maintains even λ is very large. Thus the mean values obtained maintain the same with the increase of λ, which is result expected.

1.2.2. b

From the format of the general function L, it can be seen that the loss is convex. Thus there is only one minimum point in L. With two variables m_- and m_+, a 3D figure can be drawn for L as well as the contour lines for L. To make the figure elegant, the data given in Section 1.2.3 is used. The contour lines are shown in Figure 3, and the 3D figure is shown in Figure 4.

1.2.3. c

In this case, the regularization parameter should be large enough according to Section 1.2.1. After several times of test, λ is kept 6, which is large enough to obtain the optimal solution. And the optimal solution

$$(m_-, m_+) = (0.5, 0.5)$$

.

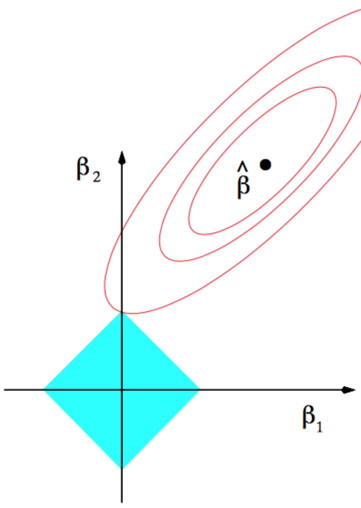
The code for plotting figures and calculating mean values is in Appendix B.

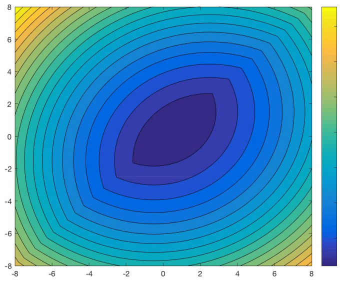
2. Some Programming & Some Experimenting

2.1. Question 3

2.1.1. a

In this case, I use gradient descent to find the mean values. Since this is a two-class digit classification task in 64 dimensions, L can be regard as a function of 128 variables.





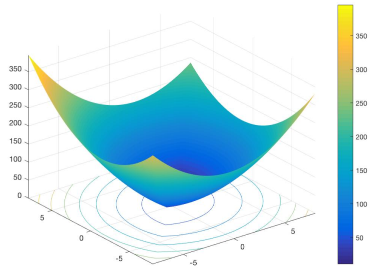


Figure 2: The "classic" illustration comparing lasso and ridge constraints. From Chapter 3 of Hastie et al.(2009)

Figure 3: The contour lines for the general function L

Figure 4: The 3D image for the function L

- Step 1: I assign these variables into two 64-digit-length vectors m_- and m_+ with a random start point.
- Step 2: Then I calculate the loss function with the initial mean, the gradient of all the mean. And update the mean to the opposite direction if their gradient with certain learning rate. Then I calculate the new loss with updated mean and compare the difference between the two loss values.
- Step 3: Loop in Step 2 until the difference is smaller than a certain termination error.

At last, the expected mean can be obtained. During this process, the learning rate and the λ should be adjusted. When λ is 0, the mean obtained does not change with the adjustment of learning rate. When λ is very large, the first part of loss function can be ignored. Thus it becomes

$$L = \lambda||m_- - m_+||_1$$

To obtain the minimum value, m_- and m_+ are expected to be almost the same. Thus when adjusting the learning rate of a large λ, I need to consider whether m_- and m_+ obtained are as expected. Otherwise, if the learning rate is too large or too small, wrong mean may be obtained.

The code to implement gradient descent is in Appendix C.

2.1.2. b

The two solution mean images I find for $\lambda = 0$ are shown in Figure 5 and Figure 6. The two solution mean images for a large λ are shown in Figure 7 and Figure 8. Here we can tell from these two images that m_- and m_+ is almost the same with a large λ, which is the result expected. Since λ is very large in this case, the learning rate should be proper to find the mean. Here the learning rate is 0.00000001. In addition, it is better to use a dynamic learning rate in this case to let the learning rate decrease in every round of calculation. Or to use the optimized learning rate α that satisfied

$$h`(\alpha) = \frac{\partial L(m + \alpha * dm)}{\partial(m + \alpha * dm)} dm$$

Here m is the mean, which is the variable of loss function L, and dm is the derivative of m. Since with adjustment of learning rate and λ by myself still can find the expected result of mean, I do not use this method to control learning rate in every round.

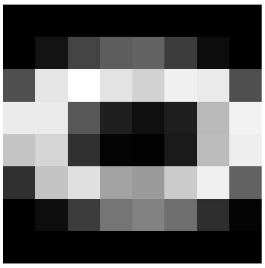


Figure 5: Image for m_- with $\lambda = 0$

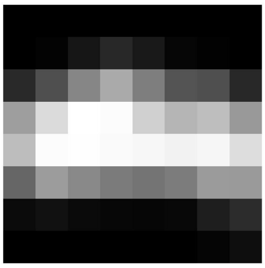


Figure 6: Image for m_+ with $\lambda = 0$

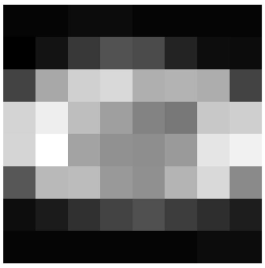


Figure 7: Image for m_- with a large $\lambda = 1250000$

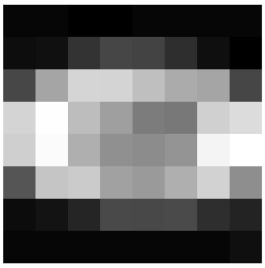


Figure 8: Image for m_+ with a large $\lambda = 1250000$

A. Code for Question 1

```
% Question 1 (a)
m = -4:0.01:4; % range of m+
l0 = (-1-m).^2 + (1-m).^2 + 0*abs(1-m); % loss function with lambda 0
% plot all the loss function with lambda {0,2,4,6}
figure
plot(m,l0)
title('Loss function as a function of m+')
xlabel('m+')
ylabel('loss')
% Question 1 (b)
% the minimizer and the minimum values
m0 = min(l0);
m0_er = m(find(l0==m0));
% the point where the derivative equals 0
dx = 1e-3;
xi = -4:dx:4;
yi0 = interp1(m, l0, xi);
dyi0 = [0 diff(yi0)]/dx;
dy0 = interp1(xi, dyi0, m);
min_dy0 = min(abs(dy0(2:801)));
der0_pos1 = find(dy0==min_dy0);
m(der0_pos1)
l0(der0_pos1)
```

B. Code for Question 2

```
%Question 2
m_minus = -8:0.01:8;
m_plus = -8:0.01:8;
[X,Y] = meshgrid(m_minus, m_plus);
L = (-1-Y).^2 + (1-Y).^2 + (3-X).^2 + (-1-X).^2 + 6*abs(X-Y);
% contour figure
figure
contourf(X,Y,L,20)
colorbar
% 3D figure
figure
surfc(X,Y,L)
colorbar
shading interp
% mean values
m = min(min(L));
[x,y] = find(L==m);
m_minus(x)
m_plus(y)
```

C. Code for Question 3

```
M_m = zeros(1,size(X,2)); % initial values of m-
M_p = zeros(1,size(X,2)); % initial values of m+
lambda = 1250000;
m = 0.00000001; % learning rate
change = 1; % initial change
k = 0; % loop number
while min(abs(change)) > 0.00000001
    L = sum((X(1:554,:)-repmat(M_m,554,1)).^2) + sum((X(555:1125,:)...
    -repmat(M_p,571,1)).^2) + lambda*abs(M_m-M_p);
    dm = 554*2*M_m - sum(2*X(1:554,:))+lambda*sign(M_m-M_p); % gradient ...
        of m-
    dp = 571*2*M_p - sum(2*X(555:1125,:))-lambda*sign(M_m-M_p); % ...
        gradient of m+
    M_m = M_m - m * dm; % update m- to the opposite direction of its ...
        gradient
    M_p = M_p - m * dp; % update m+ to the opposite direction of its ...
        gradient
    L_temp = sum((X(1:554,:)-repmat(M_m,554,1)).^2) + sum((X(555:1125,:)...
    -repmat(M_p,571,1)).^2) + lambda*abs(M_m-M_p);
    change = L - L_temp;
    k = k + 1;
end

img01 = reshape(M_m,[8, 8]);
img1 = mat2gray(img01);
figure
imshow(img1, 'InitialMagnification', 'fit');

img02 = reshape(M_p,[8, 8]);
img2 = mat2gray(img02);
figure
imshow(img2, 'InitialMagnification', 'fit');
```