IN4320 Machine Learning Exercise: Regularization & Sparsity

Yadong Li(4608283)

February 25, 2017

1 Question 1

1.1 Draw the loss function

The loss function for this particular case is

$$L(m_+) = \sum_i^{n^+}(x_i^+ - m_+)^2 + \sum_i^{n^-}(x_i^- - 1)^2 + \lambda|1 - m_+|$$

where x_i^+ *represents* x *in* $+$ *class,* x_i^- *represents* x *in* $-$ *class.*

From the equation above, we can tell that the second part of the sum is an unknown constant term, since we don't have any knowledge about the observations for $-$ class. To make the loss function simpler, we call this constant term as C:

$$L(m_+) = 2m_+^2 + 2 + \lambda|1 - m_+| + C$$

Figure 1 shows the loss function as a function of m_+ for all $\lambda \in \{0, 2, 4, 6\}$.

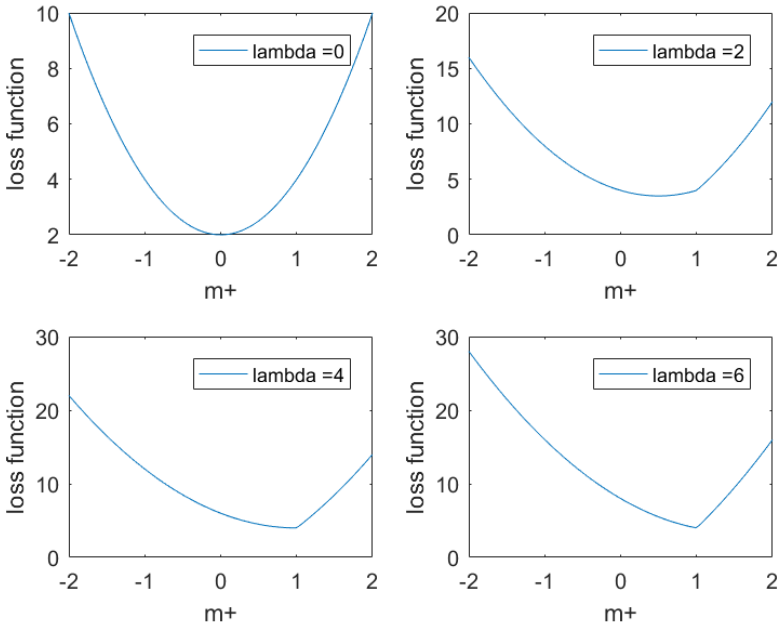


Figure 1: Four plots for different value of λ.

1.2 Derive the minimizer and minimum values

The loss function mentioned above is a continuous function, but sometimes it has a non-differentiable point. What's more, this function has a global minimum. Based on the analysis above, we can conclude that the minimum point can be derived by setting the derivative to zero, or the non-differentiable point is the minimum. We can check these two conditions to get the minimum.

1.2.1 $\lambda = 0$

When $\lambda = 0$, the loss function becomes:

$$L(m_+) = 2m_+^2 + 2 + C$$

This becomes a simple quadratic function. We can set the derivative to zero and we get:

$$\frac{dL(m_+)}{dm_+} = 4m_+ = 0 \quad \Rightarrow m_+ = 0$$

The minimum value is $2+C$.

1.2.2 $\lambda = 2$

When $\lambda = 2$, the loss function becomes:

$$L(m_+) = 2m_+^2 + 2 + 2|1 - m_+| + C$$

We can derive the minimizer by classified discussion. First, if $m_+ < 1$, we can set the derivative to zero and got $L(\frac{1}{2}) = 3.5 + C$. Second, if $m_+ = 1$, $L(1) = 4 + C$. Finally, if $m_+ > 1$, since the derivative can't be zero, there can not be a minimal point. To conclude, the minimum here is $L(\frac{1}{2}) = 3.5 + C$.

1.2.3 $\lambda = 4$

When $\lambda = 2$, the loss function becomes:

$$L(m_+) = 2m_+^2 + 2 + 4|1 - m_+| + C$$

We can derive the minimizer by classified discussion. First, if $m_+ < 1$, the derivative can't be zero. Second, if $m_+ = 1$, $L(1) = 4 + C$. Finally, if $m_+ > 1$, since the derivative can't be zero, there can not be a minimal point. To conclude, the minimum here is $L(1) = 4 + C$.

1.2.4 $\lambda = 6$

This is a very similar case as $\lambda = 4$. To conclude, the minimum here is $L(1) = 4 + C$.

2 Question 2

We now consider the setting in which both means have to be determined through a minimization of the loss.

2.1 Question $2a$

The regularizer $\lambda||m_- - m_+||_1$ tries to penalize large $L1$ distance between m_- and m_+. If λ gets larger and larger, what eventually will happen is that the $L1$ distance between m_- and m_+ will get smaller and smaller and finally become zero, which means that m_- equals to m_+.

2.2 Question $2b$

Figure 2 shows the mesh plot for a 1-dimensional dataset. From this example plot, we can tell that the loss is convex, so it has an global minimal point. The contour lines for the general function L are also convex. What's more, the contours consist of the concatenation of two basic quadratic curves when $\lambda \neq 0$.This is shown in figure 3a, where we can clearly find two quadratics curves separated by the red line($m_- = m_+$). When $\lambda = 0$, the regularizer disappears and the contours become ellipses, as shown in figure 3b.

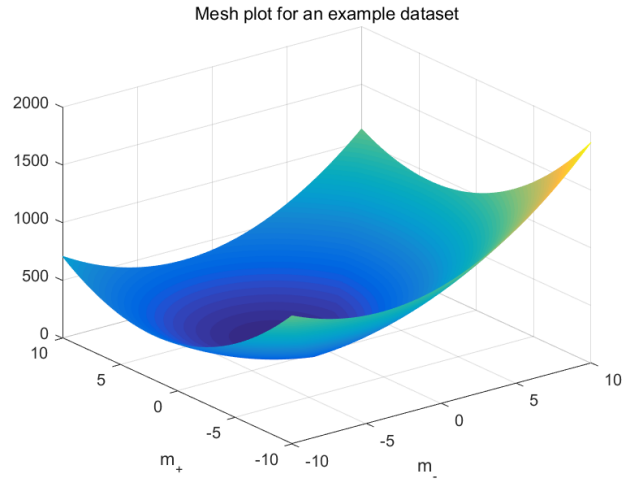
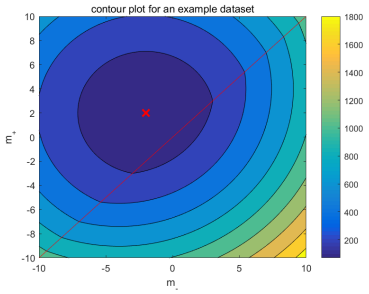
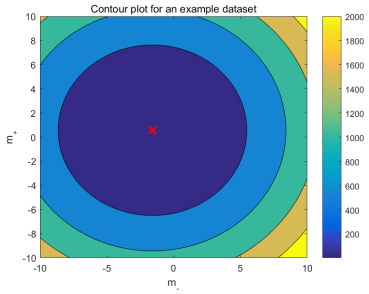


Figure 2: Contour plot for an example dataset.



(a) λ is 10



(b) λ is 0

Figure 3: Contour plot for an example dataset. The red cross represents the minimal point. The red line in 3a represents when $m_- = m_+$.

2.3 Question 2*c*

For the + class we have observations $x_1 = -1$,$x_2 = 1$. For the − class we have observations $x_3 = 3$,$x_4 = -1$. If we set a large enough λ, the regularizer will penalize any difference between m_- and m_+. Thus m_- and m_+ will be identical. They will be the global mean of these four points, which is $\frac{1}{2}$.

$$m_- = m_+ = \frac{1}{2}$$

3 Implementation of coordinate descent and gradient descent

Noticed that the loss function is a multi variate function, and because of the L1 norm term, it has some non-differentiable point. To solve this, I implemented two methods: coordinate descent and gradient descent.

3.1 Coordinate descent

The intuition behind coordinate descent is that often it is hard to find minimum for all coordinates, but easy for each coordinate. So we can pick a coordinate each time, and try to optimize loss function one coordinate at a time. To optimize one coordinate, we just fix all other coordinates and take partial *w.r.t.* m_j. To calculate the partial of L_1 penalty term, I use subgradients to generalize gradients to non-differentiable points. And then I set subgradient to 0 to get the optimal solution.

For instance, the optimal solution for m_- at coordinate j is:

$$m_{-j} = \begin{cases} \frac{\lambda}{2N_-} + \bar{x}_{-j} & \bar{x}_{-j} < m_{+j}, \\ \frac{N_-\bar{x}_{-j} + N_+\bar{x}_{+j}}{N_- + N_-} & \bar{x}_{-j} < m_{+j} \\ -\frac{\lambda}{2N_-} + \bar{x}_j & \bar{x}_{-j} > m_{+j}. \end{cases}$$

Another issue worth considering is converge criteria. For convex problems, the steps to take will become smaller and smaller.Thus, we can measure size of steps taken in a full loop over all features, and stop when max step is less than a ϵ, where ϵ can be set to be a small number. Finally, the algorithm can be described as follows:

1. Initialize m_+ and m_-
2. While not converged:

   pick a coordinate j

   $m_j \leftarrow argmin\ loss(m_j)$

To ensure the correctness of this algorithm, I've made a contour graph of 1-Dimensional case to visualize each steps it takes to find the optimal. In figure 4, the red cross indicates the optimal, and the green crosses represents each step of this coordinate descent algorithm. From the figure we can tell that it converges very fast, just 2 steps.

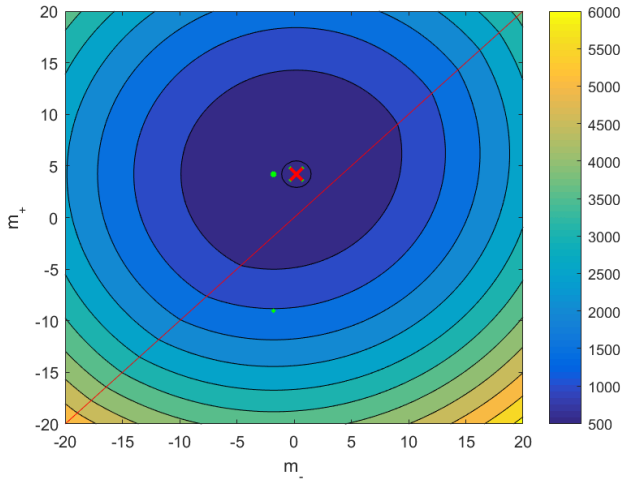


Figure 4: Visualization for coordinate descent.

3.2 Gradient descent

I also implemented a gradient descent algorithm. The intuition behind this algorithm is that if each time I take a small step towards the opposite direction of the gradient, and finally I can reach the minimal. The tricky part is to calculate gradients for non-differentiable points and to choose the step size. A simple way to generate gradients is to use sign function to generate gradients for L_1 part. But it causes some problems, too. If the step size is fixed, it is very likely that it would bounce around and never reach the optimal. To solve this, I tried to use a sigmoid function instead to represent gradients of L_1 term: $g(x) = \frac{2}{1+\exp(-x)} - 1$. This sigmoid function ensures that the partial derivative approaches 0 smoothly as we approach the bottom of our convex function. As we approach a local minimum, gradient descent can automatically take smaller steps. The visualization for this algorithm is shown in figure 5.

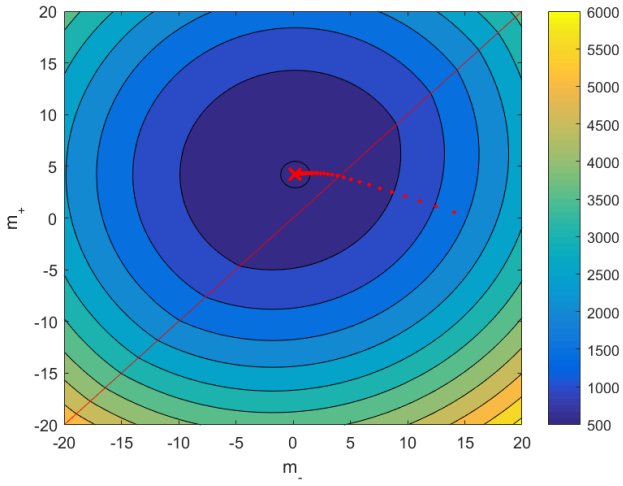


Figure 5: Visualization for gradient descent.

3.3 Plot the results

If $\lambda = 0$, the result is shown in figure 6. Figure 6a shows the reconstructed mean for class $-$ and figure 6b shows the reconstructed mean for class $+$.

(a) m_- for class $-$ (b) m_+ for class $+$

Figure 6: Reconstructed solutions for $\lambda = 0$

When λ is large enough, for example, $\lambda = 1,300,000$, the result is shown in figure 7. Figure 7a shows the reconstructed mean for class $-$ and figure 7b shows the reconstructed mean for class $+$. We can tell that when λ is large enough, the two mean figures are identical and become the "average" of all figures both in class $-$ and class $+$.

(a) m_- for class $-$ (b) m_+ for class $+$

Figure 7: Reconstructed solutions for λ is large enough