IN4320 Machine Learning

Exercises Regularization and Sparsity

Author:
Milan Niestijl, 4311728

February 27, 2017

Theory

In these exercises, a regularized version of the nearest mean classifier (NMC) is considered for two-class data. We first develop some theory, which will help us answer the questions later on.
Mathematically, the problem can be formulated as follows: Given $\lambda \in \mathbb{R}$, N d-dimensional feature vectors $x_i \in \mathbb{R}^d$, $i \in \{1, 2, ..., N\}$ and their label $y_i \in +, -$, find $m_-, m_+ \in \mathbb{R}^d$ such that the loss function L is minimized:

$$L(m_-, m_+) = \sum_{i=1}^{N} \|x_i - m_{y_i}\|_2^2 + \lambda \|m_- - m_+\|_1 \tag{1}$$

The optimal solution is denoted by $(\hat{m}_-, \hat{m}_+)$. That is,

$$(\hat{m}_-, \hat{m}_+) = \underset{m_-, m_+}{\arg\min}\, L(m_-, m_+) \tag{2}$$

In order to find the optimal solution, we note that the loss function can be decomposed in the following manner:

$$\begin{aligned} L(m_-, m_+) &= \sum_{i=1}^{N} \|x_i - m_-\|_2^2 \mathbb{1}_{\{y_i=-\}} + \sum_{i=1}^{N} \|x_i - m_+\|_2^2 \mathbb{1}_{\{y_i=+\}} + \lambda \|m_- - m_+\|_1 \\ &= L_-(m_-) + L_+(m_+) + \lambda L_1(m_-, m_+) \end{aligned} \tag{3}$$

Where:

$$\begin{aligned} L_-(m_-) &= \sum_{i=1}^{N} \|x_i - m_-\|_2^2 \mathbb{1}_{\{y_i=-\}} \\ L_+(m_+) &= \sum_{i=1}^{N} \|x_i - m_+\|_2^2 \mathbb{1}_{\{y_i=+\}} \\ L_1(m_-, m_+) &= \|m_- - m_+\|_1 \end{aligned} \tag{4}$$

We now proceed to find the optimal solution. Let $j \in \{1, 2, ..., d\}$. In order to calculate the partial derivatives $\frac{\partial L}{\partial (m_-)_j}$ and $\frac{\partial L}{\partial (m_+)_j}$, we assume $(m_-)_j \neq (m_+)_j$. Let n, $k > 0$ be the number of $-$ and $+$ labels, respectively. We have:

$$\begin{aligned} \frac{\partial L}{\partial (m_-)_j} &= 2nm_- - 2\sum_{i=1}^{N} x_{ij} \mathbb{1}_{\{y_i=-\}} + \frac{\lambda (m_- - m_+)_j}{|(m_- - m_+)_j|} \\ \frac{\partial L}{\partial (m_+)_j} &= 2km_+ - 2\sum_{i=1}^{N} x_{ij} \mathbb{1}_{\{y_i=+\}} - \frac{\lambda (m_- - m_+)_j}{|(m_- - m_+)_j|} \end{aligned} \tag{5}$$

Setting the partial derivatives to zero and rearranging the terms, we find:

$$\begin{aligned} (m_-)_j > (m_+)_j \implies \begin{cases} (\hat{m}_-)_j = \overline{(m_-)_j} - \frac{1}{2n}\lambda \\ (\hat{m}_+)_j = \overline{(m_+)_j} + \frac{1}{2k}\lambda \end{cases} \\ (m_-)_j < (m_+)_j \implies \begin{cases} (\hat{m}_-)_j = \overline{(m_-)_j} + \frac{1}{2n}\lambda \\ (\hat{m}_+)_j = \overline{(m_+)_j} - \frac{1}{2k}\lambda \end{cases} \end{aligned} \tag{6}$$

Where $\overline{(m_-)_j} = \frac{1}{n}\sum_{i=1}^{N} x_{ij} \mathbb{1}_{\{y_i=-\}}$ and $\overline{(m_+)_j} = \frac{1}{k}\sum_{i=1}^{N} x_{ij} \mathbb{1}_{\{y_i=+\}}$.

Thus, for both partial derivatives to be simultaneously to be zero, we require $\hat{m}_- \geq \hat{m}_+$ or $\hat{m}_- \leq \hat{m}_+$,

respectively, in which case the optimal solution $(\hat{m}_-, \hat{m}_+)_j$ is given by equation 6. This requirement can be reformulated as:

$$|\overline{(m_-)_j} - \overline{(m_+)_j}| \geq \lambda \left(\frac{1}{2n} + \frac{1}{2k} \right) \tag{7}$$

If this condition is not met, then there is no point at which both partial derivatives are simultaneously zero. We proceed and assume no such point exists. As the loss function L is convex, a (global) minimum does exist. Since there is no point at which the gradient is zero, it follows that the derivative is not defined at the minimum. Hence, we must have $(m_-)_j = (m_+)_j =: (m)_j$, so that the contribution of the j^{th} coordinate to the loss function is just $\sum_{i=1}^{N} |(x_i - m)_j|^2$. This is clearly minimized by the average of the j^{th} coordinate of the training data:

$$(\hat{m})_j = (\hat{m}_-)_j = (\hat{m}_+)_j = \frac{1}{N} \sum_{i=1}^{N} x_{ij} \tag{8}$$

We conclude that if condition 7 is met, then $(\hat{m}_-, \hat{m}_+)_j$ is given by equation 6. If on the other hand the condition is not met, then it is given by equation 8.

Exercises

1.

a

The loss function $L(m_+)$ is plotted for $\lambda \in \{0, 2, 4, 6\}$ in figure 1

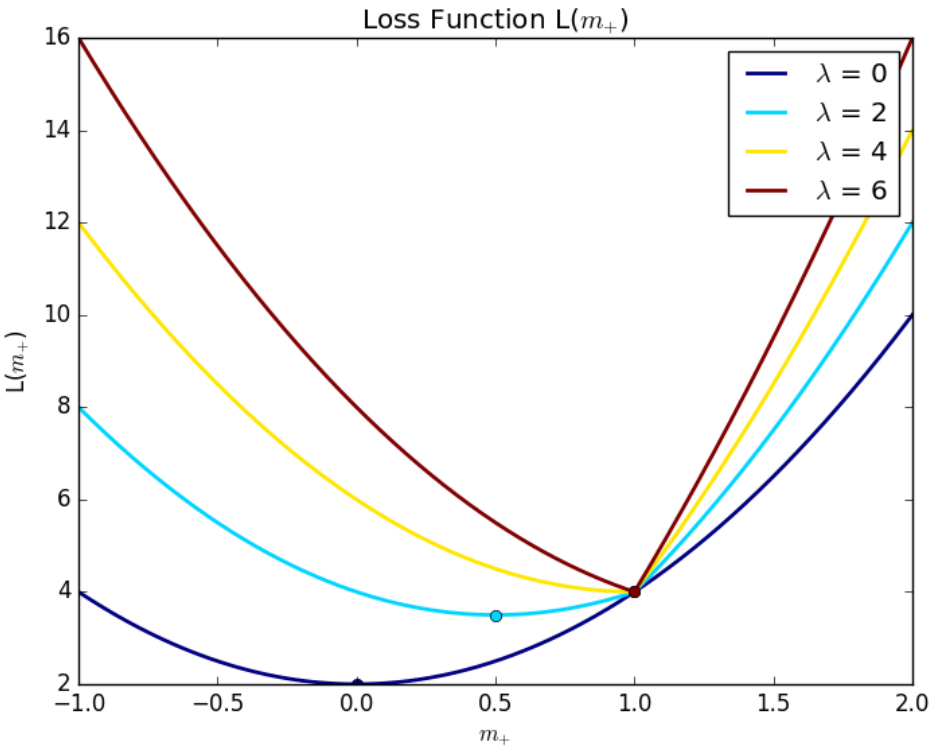

Figure 1: Loss function $L(m_+)$ for various values of λ, using $d = 1$, $m_- = 1, x_1 = 1, x_2 = -1$ and $y_1 = y_2 = +$. The dots indicate the minimum of the function.

b

Fixing $m_- = 1$, $\overline{m_+} = 0$, $n = 0$ and $k = 2$, condition 7 reduces to $\lambda < 4$. Hence for $\lambda \in \{0, 2\}$, there exists a point where the derivative equals zero, namely the point $\hat{m_+} = \frac{\lambda}{4}$. This point is thus the minimizer for $\lambda \in \{0, 2\}$. For $\lambda \in \{4, 6\}$, there is no point where the derivative equals zero, but the minimizer must satisfy $m_+ = m_-$, hence $m_+ = 1$. The minimum values are now easily calculated using the loss function. The result is shown in table 1.

Table 1: Overview of the minimizer and minimum value of the loss function $L(m_+)$ for various λ.

λ	$\hat{m_+}$	$L(\hat{m_+})$
0	0	2
2	0.5	3.5
4	1	4
6	1	4

2.

a

In general, the regularizer induces a bias towards the line $m_- = m_+$. That is, for increasing λ the minimizer moves along the line that projects the unregularized ($\lambda = 0$) minimizer onto the line $m_- = m_+$. Thus, in the limiting behaviour $\lambda \to \infty$, the minimizer is given by this projection onto the line $m_- = m_+$.

b

A contour line for the general loss function L consists of a concatenation of two ellipses that intersect on the line $m_- = m_+$. A plot is shown in figure 2. It can be seen that the analytic minimizers, which are given by equations 6 or 8, agree with the contour plot. Furthermore, the contour lines are "pulled" towards the line $m_- = m_+$ for increasing λ.

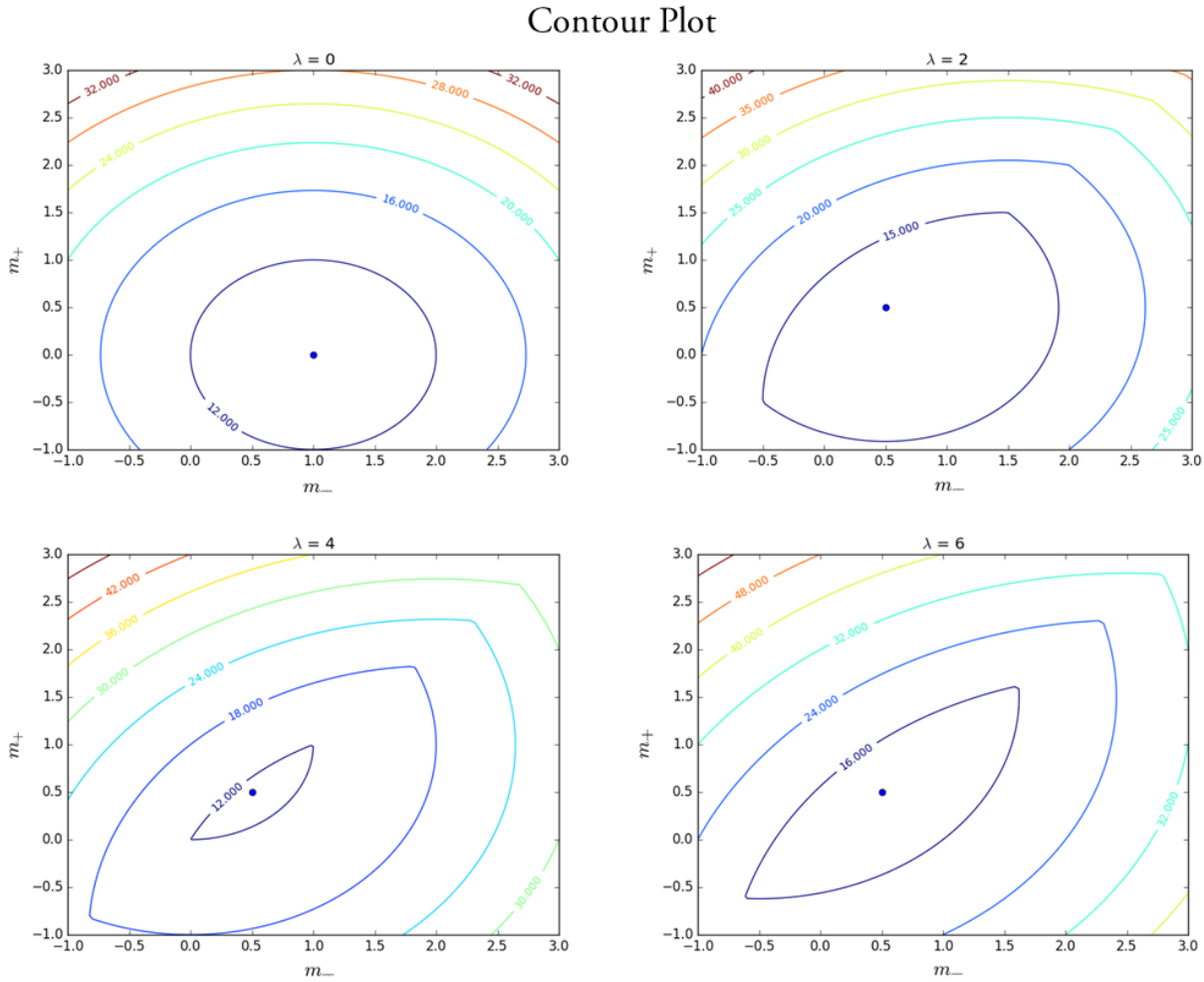

Figure 2: Contour plot of $L(m_-, m_+)$ for various values of λ in the one-dimensional case. As data, $x = [-1, 1, 3, -1]$ and $y = [1, 1, 0, 0]$ was used. The dots indicate the analytic minimizer of the loss function.

c

The orthogonal projection of the point $(1, 0)$ onto the line $m_- = m_+$ is given by $\left(\frac{1}{2}, \frac{1}{2}\right)$. Hence, for large enough λ, the solution $(\hat{m}_-, \hat{m}_+)$ will equal $\left(\frac{1}{2}, \frac{1}{2}\right)$.

3.

a

The main ingredients of the optimization routine are explained in the theory part of this report. The actual implementation first separates the input data according to their class label, then checks using condition 7 for each dimension j whether or not a point exists where both partial derivatives $\frac{\partial L}{\partial (m_-)_j}$ and $\frac{\partial L}{\partial (m_+)_j}$ equal zero. If so, it calculates $(\hat{m}_-, \hat{m}_+)_j$ according to equation 6. If not, then $(\hat{m}_-, \hat{m}_+)_j$ is calculated using equation 8 instead. The (Python) code can be found at the end of the document.

b

A plot for the two solution mean images for $\lambda = 0$ and λ large can be found in figure 3. It can be seen that for large λ, the two image coincide, which is expected, as is explained in section **2.a**.

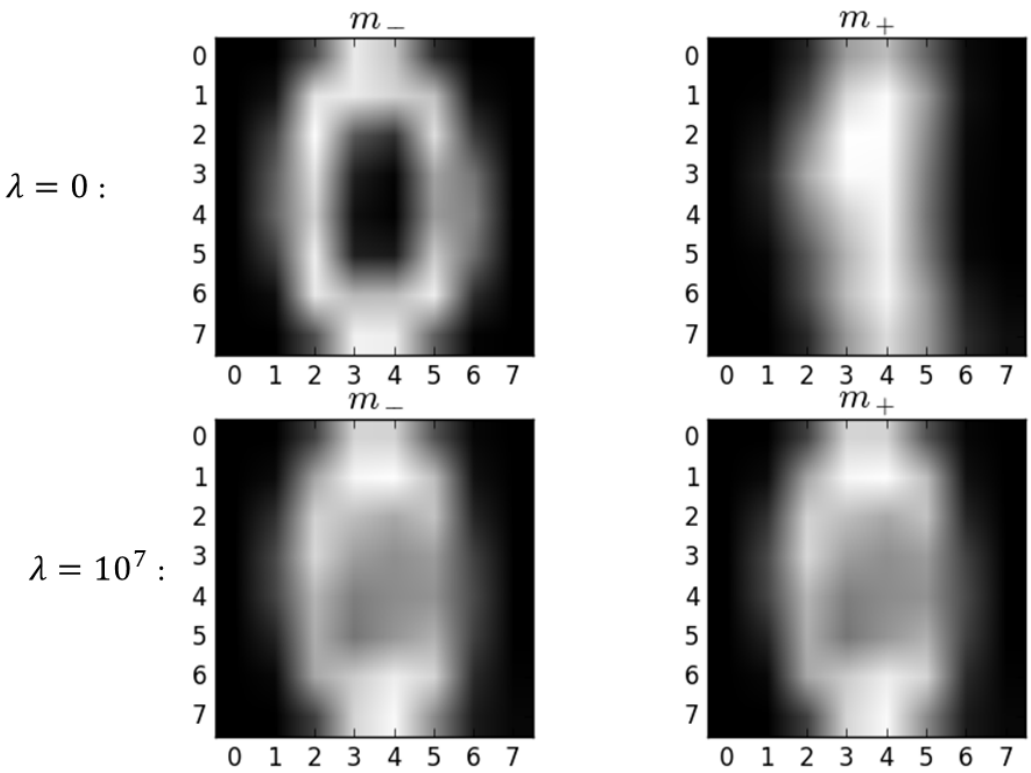

Figure 3: Image of the solution $(\hat{m}_-, \hat{m}_+)$ for $\lambda = 0$ and λ large.

Code:

```
class NMC():
    def __init__(self, Lambda):
        self.Lambda = Lambda

    # X ~ d*N, where d is the dimension of the data and N is the number of training samples.
    # y ~ N, classifying the i-th training example as 1 or 0, respectively.
    def fit(self, X, y):
        X0 = X[:,np.where(y==0)[0]]
        X1 = X[:,np.where(y==1)[0]]
        X0Mean = X0.mean(1)
        X1Mean = X1.mean(1)
        d = X.shape[0]
        n0 = X0.shape[1]
        n1 = X1.shape[1]
        M = np.zeros([d,2])
        for j in range(0,d):
            lambdaTerm = self.Lambda*(1/(2*n0) + 1/(2*n1))
            if (X0Mean[j] - X1Mean[j] >= lambdaTerm):
                M[j,0] = X0Mean[j] - self.Lambda/(2*n0)
                M[j,1] = X1Mean[j] + self.Lambda/(2*n1)
            elif (X1Mean[j] - X0Mean[j] >= lambdaTerm):
                M[j,0] = X0Mean[j] + self.Lambda/(2*n0)
                M[j,1] = X1Mean[j] - self.Lambda/(2*n1)
            else:
                M[j,[0,1]] = X[j,:].mean()
        self.M = M
```