习题二

141242006, 袁帅, 141242006@smail.nju.edu.cn

2017 年 4 月 12 日

1 [10pts] Lagrange Multiplier Methods

请通过拉格朗日乘子法 (可参见教材附录 B.1) 证明《机器学习》教材中式 (3.36) 与式 (3.37) 等价。即下面公式(1.1)与(1.2)等价。

$$\begin{aligned} \min_{\mathbf{w}} \quad & -\mathbf{w}^{\mathrm{T}}\mathbf{S}_b\mathbf{w} \\ \text{s.t.} \quad & \mathbf{w}^{\mathrm{T}}\mathbf{S}_w\mathbf{w} = 1 \end{aligned} \tag{1.1}$$

$$\mathbf{S}_b\mathbf{w} = \lambda\mathbf{S}_w\mathbf{w} \tag{1.2}$$

Proof. Using Lagrange multipliers, we have the Lagrange function

$$L(\boldsymbol{w}, \lambda) = -\boldsymbol{w}^T\boldsymbol{S}_b\boldsymbol{w} + \lambda(\boldsymbol{w}^T\boldsymbol{S}_w\boldsymbol{w} - 1). \tag{1.3}$$

Therefore, the optimal parameters $\boldsymbol{w}$ and λ should satisfy $\frac{\partial L(\boldsymbol{w},\lambda)}{\boldsymbol{w}} = -2\boldsymbol{S}_b\boldsymbol{w} + 2\lambda\boldsymbol{S}_w\boldsymbol{w} = 0$ and $\frac{\partial L(\boldsymbol{w},\lambda)}{\lambda} = \boldsymbol{w}^T\boldsymbol{S}_w\boldsymbol{w} - 1 = 0$, which immediately yields $\boldsymbol{S}_b\boldsymbol{w} = \lambda\boldsymbol{S}_w\boldsymbol{w}$, with $\boldsymbol{w}^T\boldsymbol{S}_w\boldsymbol{w} = 1$. □

2 [20pts] Multi-Class Logistic Regression

教材的章节 3.3 介绍了对数几率回归解决二分类问题的具体做法。假定现在的任务不再是二分类问题，而是多分类问题，其中 $y \in \{1, 2 \dots, K\}$。请将对数几率回归算法拓展到该多分类问题。

(1) **[10pts]** 给出该对率回归模型的"对数似然"(log-likelihood);

(2) **[10pts]** 计算出该"对数似然"的梯度。

提示 1：假设该多分类问题满足如下 $K-1$ 个对数几率，

$$\begin{aligned} \ln\frac{p(y=1|\mathbf{x})}{p(y=K|\mathbf{x})} &= \mathbf{w}_1^{\mathrm{T}}\mathbf{x} + b_1 \\ \ln\frac{p(y=2|\mathbf{x})}{p(y=K|\mathbf{x})} &= \mathbf{w}_2^{\mathrm{T}}\mathbf{x} + b_2 \\ &\cdots \\ \ln\frac{p(y=K-1|\mathbf{x})}{p(y=K|\mathbf{x})} &= \mathbf{w}_{K-1}^{\mathrm{T}}\mathbf{x} + b_{K-1} \end{aligned}$$

提示 2：定义指示函数 $\mathbb{I}(\cdot)$，

$$\mathbb{I}(y=j)=\begin{cases}1 & \text{若 } y \text{ 等于 } j\\ 0 & \text{若 } y \text{ 不等于 } j\end{cases}$$

Solution.

(1). For the sake of simplicity, we define $\hat{\boldsymbol{x}}=(\boldsymbol{x};1)$ and the parameters $\boldsymbol{\beta}_i=(\boldsymbol{w}_i;b_i)$ for all $i\in\{1,2,...,K-1\}$. By doing that, we can rewrite the log odds as

$$\ln\frac{p(y=i|\boldsymbol{x})}{p(y=K|\boldsymbol{x})}=\boldsymbol{w}_i^T\boldsymbol{x}+b_i=\boldsymbol{\beta}_i^T\hat{\boldsymbol{x}}, \tag{2.1}$$

for $i\in\{1,2,...,K-1\}$.

According to Law of total probability, i.e. $\sum_{i=1}^K p(y=i|\boldsymbol{x})=1$, by Eq.(2.1), we can derive the posterior probability expressions as

$$\begin{aligned}p(y=i|\boldsymbol{x};\boldsymbol{\beta}) &= \frac{e^{\boldsymbol{\beta}_i^T\hat{\boldsymbol{x}}}}{1+\sum_{j=1}^{K-1}e^{\boldsymbol{\beta}_j^T\hat{\boldsymbol{x}}}}, \quad \text{for } i\in\{1,2,...,K-1\}, & (2.2)\\ p(y=K|\boldsymbol{x};\boldsymbol{\beta}) &= \frac{1}{1+\sum_{j=1}^{K-1}e^{\boldsymbol{\beta}_j^T\hat{\boldsymbol{x}}}}. & (2.3)\end{aligned}$$

The log-likelihood is defined as $\ell(\boldsymbol{\beta})=\ln\Pi_{i=1}^m p(y_i|\boldsymbol{x_i})=\sum_{i=1}^m\ln p(y_i|\boldsymbol{x}_i;\boldsymbol{\beta})$, where m is the number of data points and $\boldsymbol{\beta}$ denotes all parameters $\boldsymbol{\beta}_i$. Using the indicator function $\mathbb{I}(\cdot)$, we can rewrite the log-likelihood term as

$$\ln p(y_i|\boldsymbol{x}_i;\boldsymbol{\beta})=\sum_{j=1}^K\mathbb{I}(y_i=j)\cdot\ln p(y=j|\boldsymbol{x}_i;\boldsymbol{\beta}). \tag{2.4}$$

For the sake of simplicity, we denote the denominator in Eq.(2.2) as $E_i=1+\sum_{j=1}^{K-1}e^{\boldsymbol{\beta}_j^T\hat{\boldsymbol{x}}_i}$, so by plugging Eq.(2.2) and Eq.(2.3) into Eq.(2.4), we yield log-likelihood

$$\begin{aligned}\ell(\boldsymbol{\beta}) &= \sum_{i=1}^m\left(\mathbb{I}(y_i=K)\cdot(-\ln E_i)+\sum_{j=1}^{K-1}\mathbb{I}(y_i=j)\cdot(\boldsymbol{\beta}_j^T\hat{\boldsymbol{x}}_i-\ln E_i)\right)\\ &= \sum_{i=1}^m\left((-\ln E_i)\cdot\sum_{j=1}^{K}\mathbb{I}(y_i=j)+\sum_{j=1}^{K-1}\mathbb{I}(y_i=j)\cdot\boldsymbol{\beta}_j^T\hat{\boldsymbol{x}}_i\right)\\ &= \sum_{i=1}^m\left(-\ln(1+\sum_{j=1}^{K-1}e^{\boldsymbol{\beta}_j^T\hat{\boldsymbol{x}}_i})+\sum_{j=1}^{K-1}\mathbb{I}(y_i=j)\cdot\boldsymbol{\beta}_j^T\hat{\boldsymbol{x}}_i\right). & (2.5)\end{aligned}$$

(2). We can compute the gradient of $l(\boldsymbol{\beta})$ as follow:

$$\begin{aligned}\frac{\partial\ell(\boldsymbol{\beta})}{\boldsymbol{\beta}_j} &= \sum_{i=1}^m\left(-\frac{1}{E_i}\cdot\frac{\partial E_i}{\partial\boldsymbol{\beta}_j}+\mathbb{I}(y_i=j)\cdot\hat{\boldsymbol{x}}_i\right)\\ &= \sum_{i=1}^m\left(-\frac{1}{E_i}\cdot e^{\boldsymbol{\beta}_j^T\hat{\boldsymbol{x}}_i}\hat{\boldsymbol{x}}_i+\mathbb{I}(y_i=j)\cdot\hat{\boldsymbol{x}}_i\right)\\ &= \sum_{i=1}^m[-p(y=j|\boldsymbol{x};\boldsymbol{\beta})+\mathbb{I}(y_i=j)]\cdot\hat{\boldsymbol{x}}_i\\ &= \sum_{i=1}^m\hat{\boldsymbol{x}}_i[\mathbb{I}(y_i=j)-p(y=j|\boldsymbol{x};\boldsymbol{\beta})]. & (2.6)\end{aligned}$$

3 [35pts] Logistic Regression in Practice

对数几率回归 (Logistic Regression, 简称 LR) 是实际应用中非常常用的分类学习算法。

(1) **[30pts]** 请编程实现二分类的 LR, 要求采用牛顿法进行优化求解, 其更新公式可参考《机器学习》教材公式 (3.29)。详细编程题指南请参见链接: `http://lamda.nju.edu.cn/ml2017/PS2/ML2_programming.html`

(2) **[5pts]** 请简要谈谈你对本次编程实践的感想 (如过程中遇到哪些障碍以及如何解决, 对编程实践作业的建议与意见等)。

Solution.

(2). The python version installed in my device is Python 2.7, incompatible with some Python 3.6 features, so I was using MATLAB for this project. The project instructions were well-organized, clear and concise.

The only problem I met had something to do with numerical issues: the exponential function and inverse matrix operator would work poorly because of overflow or matrix singularity. As a result, I set the initial $\boldsymbol{\beta}$ as $\mathbf{0}$ so that the exponential term may not explode in the first few terminations. Another good reason to start from $\mathbf{0}$ is that the scale of data in different dimensions may not be the same (I also tried normalization, but that gives no significant progress). To address the matrix singularity problem, I computed the 2-norm condition number of $\frac{\partial^2 \ell(\boldsymbol{\beta})}{\boldsymbol{\beta}\boldsymbol{\beta}^T}$, checked if it is too large ($>10^{15}$) and terminated the for loop when necessary [1]. In experiments, I found that results after about 5 iterations may be already good enough (95-96% accuracy).

A minor suggestion for programming assignments is to specify more in detail how the program would be tested. For instance, would the test data be drawn in a similar dataset, or the program would be simply tested on a disparate dataset, e.g. one with a different dimension and different features?

4 [35pts] Linear Regression with Regularization Term

给定数据集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_m, y_m)\}$, 其中 $\mathbf{x}_i = (x_{i1}; x_{i2}; \cdots; x_{id}) \in \mathbb{R}^d$, $y_i \in \mathbb{R}$, 当我们采用线性回归模型求解时, 实际上是在求解下述优化问题:

$$\hat{\mathbf{w}}^*_{\mathbf{LS}} = \arg\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2, \tag{4.1}$$

其中, $\mathbf{y} = [y_1, \cdots, y_m]^{\mathrm{T}} \in \mathbb{R}^m, \mathbf{X} = [\mathbf{x}_1^{\mathrm{T}}; \mathbf{x}_2^{\mathrm{T}}; \cdots; \mathbf{x}_m^{\mathrm{T}}] \in \mathbb{R}^{m \times d}$, 下面的问题中, 为简化求解过程, 我们暂不考虑线性回归中的截距 (intercept)。

在实际问题中, 我们常常不会直接利用线性回归对数据进行拟合, 这是因为当样本特征很多, 而样本数相对较少时, 直接线性回归很容易陷入过拟合。为缓解过拟合问题, 常对公式(4.1)引入正则化项, 通常形式如下:

$$\hat{\mathbf{w}}^*_{\mathbf{reg}} = \arg\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\Omega(\mathbf{w}), \tag{4.2}$$

其中, $\lambda > 0$ 为正则化参数, $\Omega(\mathbf{w})$ 是正则化项, 根据模型偏好选择不同的 Ω。

下面, 假设样本特征矩阵 $\mathbf{X}$ 满足列正交性质, 即 $\mathbf{X}^{\mathrm{T}}\mathbf{X} = \mathbf{I}$, 其中 $\mathbf{I} \in \mathbb{R}^{d\times d}$ 是单位矩阵, 请回答下面的问题 (需要给出详细的求解过程):

(1) **[5pts]** 考虑线性回归问题, 即对应于公式(4.1), 请给出最优解 $\hat{\mathbf{w}}^*_{\mathbf{LS}}$ 的闭式解表达式;

(2) **[10pts]** 考虑岭回归 (ridge regression)问题, 即对应于公式(4.2)中 $\Omega(\mathbf{w}) = \|\mathbf{w}\|_2^2 = \sum_{i=1}^d w_i^2$ 时, 请给出最优解 $\hat{\mathbf{w}}^*_{\mathbf{Ridge}}$ 的闭式解表达式;

(3) **[10pts]** 考虑LASSO问题, 即对应于公式(4.2)中 $\Omega(\mathbf{w}) = \|\mathbf{w}\|_1 = \sum_{i=1}^d |w_i|$ 时, 请给出最优解 $\hat{\mathbf{w}}^*_{\mathbf{LASSO}}$ 的闭式解表达式;

(4) **[10pts]** 考虑 ℓ_0-范数正则化问题,

$$\hat{\mathbf{w}}^*_{\ell_0} = \arg\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_0, \tag{4.3}$$

其中, $\|\mathbf{w}\|_0 = \sum_{i=1}^d \mathbb{I}[w_i \neq 0]$, 即 $\|\mathbf{w}\|_0$ 表示 $\mathbf{w}$ 中非零项的个数。通常来说, 上述问题是 NP-Hard 问题, 且是非凸问题, 很难进行有效地优化得到最优解。实际上, 问题 (3) 中的 LASSO 可以视为是近些年研究者求解 ℓ_0-范数正则化的凸松弛问题。

但当假设样本特征矩阵 $\mathbf{X}$ 满足列正交性质, 即 $\mathbf{X}^{\mathrm{T}}\mathbf{X} = \mathbf{I}$ 时, ℓ_0-范数正则化问题存在闭式解。请给出最优解 $\hat{\mathbf{w}}^*_{\ell_0}$ 的闭式解表达式, 并简要说明若去除列正交性质假设后, 为什么问题会变得非常困难?

Solution.

(1). Error measurement $E_{LS}(\boldsymbol{w}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 = \frac{1}{2}(\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{y}^T\boldsymbol{X}\boldsymbol{w} + \boldsymbol{y}^T\boldsymbol{y})$, so the derivative $\frac{\partial E_{LS}(\boldsymbol{w})}{\partial \boldsymbol{w}} = \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y}$. Setting $\frac{\partial E_{LS}(\boldsymbol{w})}{\partial \boldsymbol{w}} = 0$ gives $\hat{\boldsymbol{w}}^*_{\boldsymbol{LS}} = \boldsymbol{X}^T\boldsymbol{y}$.

(2). Error measurement $E_{Ridge}(\boldsymbol{w}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$, so the derivative $\frac{\partial E_{Ridge}(\boldsymbol{w})}{\partial \boldsymbol{w}} = \boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y} + 2\lambda\boldsymbol{w}$. Setting $\frac{\partial E_{Ridge}(\boldsymbol{w})}{\partial \boldsymbol{w}} = 0$ gives $\hat{\boldsymbol{w}}^*_{\boldsymbol{Ridge}} = \frac{1}{2\lambda+1}\boldsymbol{X}^T\boldsymbol{y}$.

(3). Error measurement $E_{LASSO}(\boldsymbol{w}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_1$. Notice that the error function is non-differentiable (but still continuous) at $w_i = 0$, and that

$$\begin{aligned} E_{LASSO}(\boldsymbol{w}) &= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \boldsymbol{y}^T\boldsymbol{X}\boldsymbol{w} + \lambda\sum_{i=1}^d |w_i| + C \\ &= \sum_{i=1}^d \left(\frac{1}{2}w_i^2 - \alpha_i w_i + \lambda|w_i|\right) + C, \end{aligned} \tag{4.4}$$

where $\alpha_i = (\boldsymbol{X}^T\boldsymbol{y})_i$ denotes the ith dimension in $\boldsymbol{X}^T\boldsymbol{y}$, and $C = \frac{1}{2}\boldsymbol{y}^T\boldsymbol{y}$ is a constant. Therefore, we can optimize each dimension w_i separately, i.e. $\hat{w}_i^* = \arg\min_{w_i}\left(\frac{1}{2}w_i^2 - \alpha_i w_i + |w_i|\right)$. In this expression, $|w_i|$ is a segment function, so we can rewrite $E_{LASSO}(\boldsymbol{w})$ as

$$\frac{1}{2}w_i^2 - \alpha_i w_i + |w_i| = \begin{cases} \frac{1}{2}w_i^2 - (\alpha_i - \lambda)w_i, & w_i > 0, \\ \frac{1}{2}w_i^2 - (\alpha_i + \lambda)w_i, & w_i < 0, \end{cases} \tag{4.5}$$

which is obviously a combination of two segments of quadratic functions. Consider the following cases:

(i). $\alpha_i > \lambda$. As shown in Fig.(1), the solid lines are plots of $E_{LASSO}(\boldsymbol{w})$, composed of two quadratic segments. The two symmetric axises are at the positive side of w axis, since $\alpha_i + \lambda > \alpha_i - \lambda > 0$. Thus, the optimal $\hat{w}_i^* = \alpha_i - \lambda$.

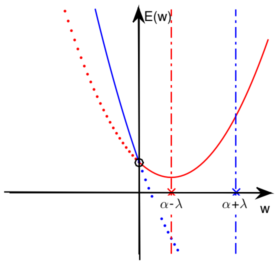

Figure 1: $\alpha_i > \lambda$

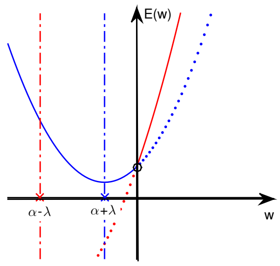

Figure 2: $\alpha_i < -\lambda$

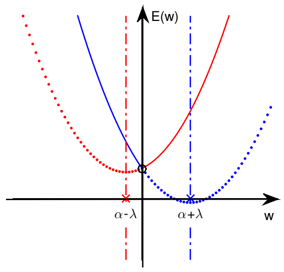

Figure 3: $-\lambda \leq \alpha_i \leq \lambda$

(ii). $\alpha_i < -\lambda$. Similarly, as shown in Fig.(2), both symmetric axises are at the negative side of w axis. The optimal $\hat{w}_i^* = \alpha_i + \lambda$.

(iii). $-\lambda \leq \alpha_i \leq \lambda$. As shown in Fig.(3), since $\alpha_i - \lambda < 0 < \alpha_i + \lambda$. The solid line shows that $E_{LASSO}(w)$ is decreasing when $w_i < 0$ and increasing when $w_i > 0$. Therefore, the optimal $\hat{w}_i^* = 0$.

Thus, the optimal $\hat{\boldsymbol{w}}^*_{\boldsymbol{LASSO}}$ could be give as $(\hat{\boldsymbol{w}}^*_{LASSO})_i = \begin{cases} (\boldsymbol{X}^T\boldsymbol{y})_i - \lambda, & (\boldsymbol{X}^T\boldsymbol{y})_i > \lambda, \\ (\boldsymbol{X}^T\boldsymbol{y})_i + \lambda, & (\boldsymbol{X}^T\boldsymbol{y})_i < -\lambda, \\ 0, & \text{otherwise}. \end{cases}$, in which $(\cdot)_i$ indicates the ith dimension.

(4). Error measurement $E_{\ell_0}(\boldsymbol{w}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \sum_{i=1}^d \mathbb{I}[w_i \neq 0]$. In this case, consider different dimensions separately, since

$$\begin{aligned} E_{\ell_0}(\boldsymbol{w}) &= \frac{1}{2}(\boldsymbol{w}^T\boldsymbol{w} - 2\boldsymbol{y}^T\boldsymbol{X}\boldsymbol{w} + \boldsymbol{y}^T\boldsymbol{y}) + \lambda \sum_{i=1}^d \mathbb{I}[w_i \neq 0] \\ &= \sum_{i=1}^d \left(\frac{1}{2}w_i^2 - \alpha_i w_i + \lambda\mathbb{I}[w_i \neq 0]\right) + C, \end{aligned} \tag{4.6}$$

where $\alpha_i = (\boldsymbol{X}^T\boldsymbol{y})_i$ denotes the ith dimension in $\boldsymbol{X}^T\boldsymbol{y}$, and $C = \frac{1}{2}\boldsymbol{y}^T\boldsymbol{y}$ is a constant. Because all dimensional components in Eq.(4.6) are independent, the optimization problem is equivalent to minimize each component. Consider the following cases:

(i). $w_i \neq 0$. We have $\hat{w}_i^* = \arg\min_{w_i} \frac{1}{2}w_i^2 - \alpha_i w_i + \lambda = \alpha_i$, with minimal value $\lambda - \frac{1}{2}\alpha_i^2$.

(ii). $w_i = 0$. The resulting component $\frac{1}{2}w_i^2 - \alpha_i w_i = 0$.

Thus, the optimal parameter would be determined by $\hat{w}_i^* = \begin{cases} \alpha_i, & \lambda - \frac{1}{2}\alpha_i^2 < 0, \\ 0, & \lambda - \frac{1}{2}\alpha_i^2 \geq 0. \end{cases}$, i.e. $(\hat{\boldsymbol{w}}^*_{\boldsymbol{\ell_0}})_i = \begin{cases} (\boldsymbol{X}^T\boldsymbol{y})_i & \frac{1}{2}(\boldsymbol{X}^T\boldsymbol{y})_i^2 > \lambda, \\ 0, & \frac{1}{2}(\boldsymbol{X}^T\boldsymbol{y})_i^2 \leq \lambda. \end{cases}$, in which $(\cdot)_i$ indicates the ith dimension.

In this problem, we can access the closed form solution thanks to the orthogonality of X, i.e. $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{I}$. Without such restriction, the problem could be hard because

- the $\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w}$ term makes different dimensions correlate, so the analysis mentioned above may not apply;

- the $\lambda\|\mathbf{w}\|_0$ term leads the point 0 to be a highly non-analytical(non-differentiable) case, making all gradient-based methods(gradient descent, Newton's method, etc.) hard to work; the function is even not continuous at the point 0, which is disastrous for all local search-based algorithms(coordinate descent, etc.);
- the error function is non-convex according to the inspection of Jensen Inequality[2].

Reference

[1] Stack Overflow, *Matlab: How to find out if a matrix is singular?*. `http://stackoverflow.com/a/13146750`.

[2] Ayush Bhatnagar. *Why is the 'L0' norm non-differentiable and non-convex?* (2016). `https://www.quora.com/Why-is-the-L0-norm-non-differentiable-and-non-convex`.