第二次作业

赵丰，2017310711

November 2, 2017

1 理论题目

P1(a) 因为 N 个点线性可分，所以存在由 $\boldsymbol{a}, b$ 确定的超平面 $\mathcal{H} = \{\boldsymbol{x} \in \mathcal{R}^d | \boldsymbol{a}^T\boldsymbol{x} + b = 0\}$, 使得 $\mathcal{H}$ 分离两类样本点，且分离的 $margin$ 为 ρ, 即满足:

$$\begin{cases} \boldsymbol{a}^T\boldsymbol{x}_i + b \le -\frac{\rho}{2} & \text{if } y_i = -1 \\ \boldsymbol{a}^T\boldsymbol{x}_i + b \ge \frac{\rho}{2} & \text{if } y_i = 1 \end{cases} \tag{1}$$

取 $t = \max\limits_{1\le i\le N} ||\boldsymbol{x}_i||$, 并令 $\hat{\boldsymbol{w}} = (\frac{2t\boldsymbol{a}}{\rho}, \frac{2tb}{\rho})$, 则可以直接验证

$$y_i\hat{\boldsymbol{w}}^T\boldsymbol{z}_i \ge 1, \forall i \in \{1, \dots, N\} \tag{2}$$

P1(b)

$$\begin{aligned} ||\boldsymbol{w}_t - \hat{\boldsymbol{w}}||^2 - ||\boldsymbol{w}_{t+1} - \hat{\boldsymbol{w}}||^2 =& 2(\boldsymbol{w}_{t+1} - \boldsymbol{w}_t) \cdot \hat{\boldsymbol{w}} + (||\boldsymbol{w}_t||^2 - ||\boldsymbol{w}_{t+1}||^2) & (3) \\ =& 2y_i(\boldsymbol{z}_i) \cdot \hat{\boldsymbol{w}} + (||\boldsymbol{w}_t||^2 - ||\boldsymbol{w}_t + y_i\boldsymbol{z}_i||^2) & (4) \\ \ge& 2 - 2y_i\boldsymbol{w}_t \cdot \boldsymbol{z}_i + y_i^2||\boldsymbol{z}||^2 & (5) \\ \ge& 1, y_i\boldsymbol{w}_t \cdot \boldsymbol{z}_i < 0, ||\boldsymbol{z}|| = 1 & (6) \end{aligned}$$

因此 $0 \le ||\boldsymbol{w}_t - \hat{\boldsymbol{w}}|| \le ||\boldsymbol{w}_0 - \hat{\boldsymbol{w}}|| - t$, 所以迭代次数 $t \le ||\boldsymbol{w}_0 - \hat{\boldsymbol{w}}||$, 向上取整即得要证的结论。

2.1(a)

$$F(\boldsymbol{W}) = \frac{1}{N}\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{W} - \boldsymbol{W}^T\boldsymbol{X}\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{X}^T\boldsymbol{W} + \boldsymbol{Y}^T\boldsymbol{Y} \tag{7}$$

令 $\frac{\partial F(\boldsymbol{W})}{\partial \boldsymbol{W}} = \boldsymbol{0}$, 解出 $\boldsymbol{W} = (\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}\boldsymbol{Y}$ 即为最优解。

2.1(b) 设 $\boldsymbol{v} \in \mathcal{R}^N$, 则 $(\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}\boldsymbol{v} \in \mathcal{R}^{d+1}$, 因此 $\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}\boldsymbol{v}$ 是 $\boldsymbol{X}^T$ 各列向量的线性组合，落在由 $\boldsymbol{X}^T$ 列向量张成的线性空间中。

为证 $\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}\boldsymbol{Y} - \boldsymbol{Y}$ 与 $\boldsymbol{X}^T$ 列向量张成的线性空间正交，只需证 $\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}\boldsymbol{Y} - \boldsymbol{Y}$ 与 $\boldsymbol{X}^T$ 各列向量垂直。即证 $(\boldsymbol{X}^T)^T\left(\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}\boldsymbol{Y} - \boldsymbol{Y}\right) = 0$，化简即得。

2.2 设

$$h_w(\boldsymbol{x}) = \frac{1}{1 + exp(-\boldsymbol{w}^T\boldsymbol{x})} \tag{8}$$

对于 Logistic Model，对数似然函数为

$$\log L(\boldsymbol{w}|\boldsymbol{x})=\sum_{i=1}^{m} y_i \log h_w(\boldsymbol{x}_i)+(1-y_i)\log(1-h_w(\boldsymbol{x}_i)) \tag{9}$$

对 $\boldsymbol{w}$ 求导得

$$\sum_{i=1}^{m}(h_w(\boldsymbol{x}_i)-y_i)\boldsymbol{x}_i=0 \tag{10}$$

上面方程关于 $h_w(\boldsymbol{x}_i)$ 的最小二乘解为 $h_w(\boldsymbol{x}_i)=y_i$，我们假设 y_i 取 ±1, 因为原数据集线性可分，所以存在 $\boldsymbol{w'}$，使得 $\boldsymbol{w'}^T\boldsymbol{x}_i>0$ 对 $y_i=1$ 成立，此时 $\frac{1}{1+exp(-\boldsymbol{w'}^T\boldsymbol{x}_i)}>\frac{1}{2}$, 增大 $\boldsymbol{w'}$ 的模长使得 $\frac{1}{1+exp(-\boldsymbol{w'}^T\boldsymbol{x}_i)}$ 更接近 1。对于 $y_i=-1$ 有类似的观察，因此 MLE 方法对 Logistic Model 给出的 $\boldsymbol{w}$ 可以沿着 $\boldsymbol{w'}$ 的方向将其模长取到任意大得到。

2.3 已知

$$\boldsymbol{m}_1=\frac{1}{M_1}\sum_{n\in C_1}\boldsymbol{x}_n, \boldsymbol{m}_2=\frac{1}{M_2}\sum_{n\in C_2}\boldsymbol{x}_n \tag{11}$$

$$\begin{aligned}
&\sum_{i=1}^{M}(\boldsymbol{w}^T\boldsymbol{x}_i+w_0-y_i)\boldsymbol{x}_i=0\\
\iff &\sum_{n\in C_1}^{M}(\boldsymbol{w}^T\boldsymbol{x}_i+w_0-y_i)\boldsymbol{x}_i+\sum_{n\in C_2}^{M}(\boldsymbol{w}^T\boldsymbol{x}_i+w_0-y_i)\boldsymbol{x}_i=0\\
\iff &\sum_{n\in C_1}^{M}(\boldsymbol{w}^T\boldsymbol{x}_i-\boldsymbol{w}^T\boldsymbol{m}-\frac{M}{M_1})\boldsymbol{x}_i+\sum_{n\in C_2}^{M}(\boldsymbol{w}^T\boldsymbol{x}_i-\boldsymbol{w}^T\boldsymbol{m}+\frac{M}{M_2})\boldsymbol{x}_i=0\\
\iff &\sum_{n\in C_1}^{M}(\boldsymbol{w}^T\boldsymbol{x}_i)\boldsymbol{x}_i+\sum_{n\in C_2}^{M}(\boldsymbol{w}^T\boldsymbol{x}_i)\boldsymbol{x}_i-(\boldsymbol{w}^T\boldsymbol{m}+\frac{M}{M_1})M_1\boldsymbol{m}_1+(-\boldsymbol{w}^T\boldsymbol{m}+\frac{M}{M_2})M_2\boldsymbol{m}_2=0\\
\iff &\underbrace{\sum_{n\in C_1}^{M}(\boldsymbol{x}_i\boldsymbol{x}_i^T)\boldsymbol{w}}_{I_1}+\underbrace{\sum_{n\in C_2}^{M}(\boldsymbol{x}_i\boldsymbol{x}_i^T)\boldsymbol{w}}_{I_2}+\underbrace{(-M_1\boldsymbol{w}^T\boldsymbol{m}-M)\boldsymbol{m}_1+(-M_2\boldsymbol{w}^T\boldsymbol{m}+M)\boldsymbol{m}_2}_{I_3}=0
\end{aligned}$$

$$\begin{aligned}
I_1&=\sum_{n\in C_1}^{M}(\boldsymbol{x}_i\boldsymbol{x}_i^T)\boldsymbol{w}\\
&=\sum_{n\in C_1}^{M}\Big(((\boldsymbol{x}_i-\boldsymbol{m}_1)(\boldsymbol{x}_i-\boldsymbol{m}_1)^T)\boldsymbol{w}+(\boldsymbol{m}_1\boldsymbol{x}_i^T)\boldsymbol{w}+(\boldsymbol{x}_i\boldsymbol{m}_1^T)\boldsymbol{w}-(\boldsymbol{m}_1\boldsymbol{m}_1^T)\boldsymbol{w}\Big)\\
&=\sum_{n\in C_1}^{M}\Big(((\boldsymbol{x}_i-\boldsymbol{m}_1)(\boldsymbol{x}_i-\boldsymbol{m}_1)^T)\boldsymbol{w}\Big)+M_1(\boldsymbol{m}_1\boldsymbol{m}_1^T)\boldsymbol{w}
\end{aligned}$$

同理

$$I_2 = \sum_{n\in C_2}^{M} \big(((\boldsymbol{x}_i - \boldsymbol{m}_2)(\boldsymbol{x}_i - \boldsymbol{m}_2)^T)\boldsymbol{w}\big) + M_2(\boldsymbol{m}_2\boldsymbol{m}_2^T)\boldsymbol{w} \tag{12}$$

因此

$$I_1 + I_2 = S_W\boldsymbol{w} + M_1(\boldsymbol{m}_1\boldsymbol{m}_1^T)\boldsymbol{w} + M_2(\boldsymbol{m}_2\boldsymbol{m}_2^T)\boldsymbol{w} \tag{13}$$

$$\begin{aligned}
I_3 =&(-M_1\boldsymbol{w}^T\boldsymbol{m} - M)\boldsymbol{m}_1 + (-M_2\boldsymbol{w}^T\boldsymbol{m} + M)\boldsymbol{m}_2\\
=&(-M_1\boldsymbol{w}^T\frac{M_1\boldsymbol{m}_1 + M_2\boldsymbol{m}_2}{M} - M)\boldsymbol{m}_1 + (-M_2\boldsymbol{w}^T\frac{M_1\boldsymbol{m}_1 + M_2\boldsymbol{m}_2}{M} + M)\boldsymbol{m}_2\\
=&(-M_1\frac{M_1\boldsymbol{m}_1\boldsymbol{m}_1^T + M_2\boldsymbol{m}_2\boldsymbol{m}_1^T}{M})\boldsymbol{w} + (-M_2\frac{M_1\boldsymbol{m}_1\boldsymbol{m}_2^T + M_2\boldsymbol{m}_2\boldsymbol{m}_2^T}{M})\boldsymbol{w} - M(\boldsymbol{m}_1 - \boldsymbol{m}_2)
\end{aligned}$$

$$\begin{aligned}
&I_1 + I_2 + I_3 = 0\\
\iff &S_W\boldsymbol{w} + M_1(\boldsymbol{m}_1\boldsymbol{m}_1^T)\boldsymbol{w} + M_2(\boldsymbol{m}_2\boldsymbol{m}_2^T)\boldsymbol{w} + (-M_1\frac{M_1\boldsymbol{m}_1\boldsymbol{m}_1^T + M_2\boldsymbol{m}_2\boldsymbol{m}_1^T}{M})\boldsymbol{w}\\
&+(-M_2\frac{M_1\boldsymbol{m}_1\boldsymbol{m}_2^T + M_2\boldsymbol{m}_2\boldsymbol{m}_2^T}{M})\boldsymbol{w} - M(\boldsymbol{m}_1 - \boldsymbol{m}_2) = 0\\
\iff &S_W\boldsymbol{w} + \frac{1}{M}(M_1(M_1+M_2)(\boldsymbol{m}_1\boldsymbol{m}_1^T) + M_2(M_1+M_2)(\boldsymbol{m}_2\boldsymbol{m}_2^T) - M_1^2\boldsymbol{m}_1\boldsymbol{m}_1^T - M_1M_2\boldsymbol{m}_2\boldsymbol{m}_1^T\\
&-M_1M_2\boldsymbol{m}_1\boldsymbol{m}_2^T - M_2^2\boldsymbol{m}_2\boldsymbol{m}_2^T)\boldsymbol{w} - M(\boldsymbol{m}_1 - \boldsymbol{m}_2) = 0\\
\iff &(S_W + \frac{M_1M_2}{M}S_B)\boldsymbol{w} - M(\boldsymbol{m}_1 - \boldsymbol{m}_2) = 0
\end{aligned}$$

3(a) 设

$$\begin{aligned}
L(\boldsymbol{u},\boldsymbol{v},\boldsymbol{w},\boldsymbol{\xi},\boldsymbol{\alpha},b) =&\frac{1}{2}||\boldsymbol{\alpha}||_2^2 + C||\boldsymbol{\xi}||_1 - \sum_{i=1}^{m} u_i\left(y_i(\sum_{j=1}^{m}\alpha_j y_j \boldsymbol{x}_i^T\boldsymbol{x}_j + b) - 1 + \xi_i\right)\\
&-\sum_{i=1}^{m} v_i\xi_i - \sum_{i=1}^{m} w_i\alpha_i
\end{aligned}\tag{14}$$

原优化问题的 Lagrange 对偶为：

$$\begin{aligned}
&\max \inf_{\boldsymbol{\xi},\boldsymbol{\alpha},b} L(\boldsymbol{u},\boldsymbol{v},\boldsymbol{w},\boldsymbol{\xi},\boldsymbol{\alpha},b)\\
&\text{s.t. } \boldsymbol{u},\boldsymbol{v},\boldsymbol{w} \geq 0
\end{aligned}\tag{15}$$

L 分别对各个变量求偏导数，得

$$\alpha_j - \sum_{i=1}^{m}(u_i y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j) - w_j = 0, j = 1, 2, \ldots, m \tag{16}$$

$$C - u_i - v_i = 0 \tag{17}$$

$$\sum_{i=1}^{m} u_i y_i = 0 \tag{18}$$

将(16)式求出的 $\boldsymbol{\alpha}$ 代入(15)式，得到 $\inf L$ 为

$$\mathcal{L} = -\frac{1}{2}\sum_{i=1}^{m} w_i^2 - \frac{1}{2}\sum_{i=1}^{m}\left(\sum_{j=1}^{m} u_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j\right)^2 - \sum_{i,j=1}^{m}(y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j) u_i w_j + \sum_{i=1}^{m} u_i \tag{19}$$

由(17)式，关于 $\boldsymbol{v}$ 的约束可以转化为 $u_i \le C$，因此(15)式化简为:

$$\begin{aligned}&\max_{\boldsymbol{u},\boldsymbol{w}} -\frac{1}{2}\sum_{i=1}^{m} w_i^2 - \frac{1}{2}\sum_{i=1}^{m}\left(\sum_{j=1}^{m} u_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j\right)^2 - \sum_{i,j=1}^{m}(y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j) u_i w_j + \sum_{i=1}^{m} u_i \\ &\text{s.t. } w_j \ge 0, 0 \le u_j \le C, \boldsymbol{u}^T \boldsymbol{y} = 0\end{aligned} \tag{20}$$

注意到对偶问题是关于 $\boldsymbol{u}, \boldsymbol{w}$ 的带有区间不等式约束的二次规划问题。$u_i u_k$ 的系数为 $(y_j \boldsymbol{x}_j)^T(\sum \boldsymbol{x}_i \boldsymbol{x}_i^T)(y_k \boldsymbol{x}_k)$, 具有内积的表达形式，因此目标函数是约束变量的凸函数，可采用凸优化的方法求解对偶问题。

3(b) 考虑关于 $\boldsymbol{\alpha}$ 的 1 范数，

$$\begin{aligned}L(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, b) =& ||\boldsymbol{\alpha}||_1 + C||\boldsymbol{\xi}||_1 - \sum_{i=1}^{m} u_i \left(y_i(\sum_{j=1}^{m} \alpha_j y_j \boldsymbol{x}_i^T \boldsymbol{x}_j + b) - 1 + \xi_i\right) \\ & - \sum_{i=1}^{m} v_i \xi_i - \sum_{i=1}^{m} w_i \alpha_i\end{aligned} \tag{21}$$

此时(16)式变为

$$1 - \sum_{i=1}^{m}(u_i y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j) - w_j = 0, j = 1, 2, \ldots, m \tag{22}$$

代入(21)式中，得对偶问题为

$$\max_{\boldsymbol{u}} \sum_{i=1}^{m} u_i$$

$$\text{s.t. } 0 \le u_j \le C, \tag{23}$$

$$\boldsymbol{u}^T \boldsymbol{y} = 0, \tag{24}$$

$$1 - \sum_{i=1}^{m}(u_i y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j) \ge 0, j = 1, 2, \ldots, m \tag{25}$$

2 Program Practice

使用 Logistic 方法和 SVM 方法分别对给定的数据进行 2 分类，

2.1 Logistic Regression

Doing: 针对给定的数据，首先采用线性归一化的方法将每一维数据压缩到 $[0,1]$ 区间上。

a: 采用 Gradient Ascent Method 而不是求解非线性方程的方法更新模型参数：迭代公式为:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha(y_i - h_w(\boldsymbol{x}_i)\boldsymbol{x}_i \tag{26}$$

具体更新时可用训练集多次迭代直到误差不再下降为止。

b: 在似然函数中加入先验（正则因子）$\frac{\lambda}{2}||\boldsymbol{w}||^2$, 其中 λ 为正则化因子。在给定 λ 的情形下，对似然函数关于 $\boldsymbol{w}$ 求梯度得非线性方程:

$$\nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w}) = \lambda\boldsymbol{w} + \sum_{i=1}^{m}(y_i - h_w(\boldsymbol{x}_i))\boldsymbol{x}_i = 0 \tag{27}$$

其中有 m 个训练数据。根据下面的 IRLS 方法的启示，可以用 Newton 迭代法求解上面的非线性方程，只不过此时以(27)为梯度，Hessian 矩阵为:

$$H = -\boldsymbol{X}\boldsymbol{R}\boldsymbol{X}^T + \lambda\boldsymbol{I} \tag{28}$$

其中 $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_m]$ 是 $n \times m$ 的矩阵, $\boldsymbol{R}$ 是对角阵，如设 $u_i = h_w(\boldsymbol{x}_i)(1 - h_w(\boldsymbol{x}_i))$，则对角元为 $R_{ii} = u_i(1 - u_i)$ H 的阶数与 $\boldsymbol{w}$ 的维数相同，通过加上一个 $\lambda\boldsymbol{I}$ 的单位阵可以改善 H 的正定性质。于是我们的迭代步为:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + (\boldsymbol{X}\boldsymbol{R}\boldsymbol{X}^T + \lambda\boldsymbol{I})^{-1}(\boldsymbol{X}(\boldsymbol{y} - \boldsymbol{u}) + \lambda\boldsymbol{w}_t) \tag{29}$$

$\boldsymbol{u}, \boldsymbol{y}$ 是 m 维的向量

我们首先把把数据集分成 5 份，选不同的 λ 代入计算，每次计算中，分 5 轮，每一轮中以 4 份训练 1 分验证，5 次结果取平均得到错误率。

Report:

a: 由于全部数据均带有标记，可以将其按照 4:1 的关系分为训练数据和测试数据，其中训练数据用于训练模型参数，然后在测试数据上进行预测。

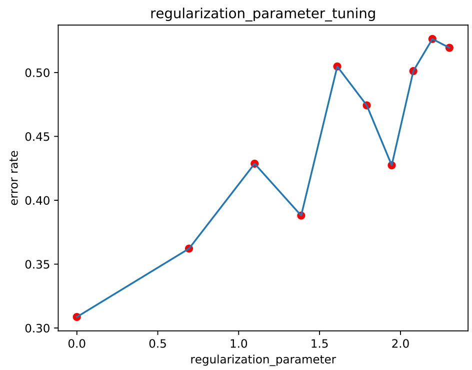
b: 对于正则化的 Logistic 回归，我们通过调整正则化参数，得到其在测试集上最小的预测误差为 0.31。并且在正则化参数接近 0 时取得，结果与 IRLS 方法的预测误差近似。

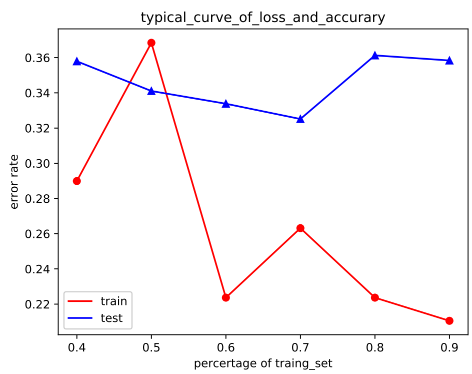
c: 以 IRLS 方法为例，我们按 $d\%$ 抽取训练集，其余数据作为测试集，以训练集的百分比为横轴，以误差为纵轴，分别画出训练出的模型在训练集和测试集上的误差曲线如下所示：

从上图可以看出，当训练集用的比重过大时，尽管在训练集上的误差下降到了 30% 以下，但在测试集上的误差却有明显上升，即出现了明显的过拟合的问题。

对于标准的随机梯度方法，我们得到测试集上的准确率为 0.32 ± 0.04, 对于 IRLS 方法，准确率为 0.31 ± 0.04, IRLS 方法略优于随机梯度法，但计算开销方面却较大。

d: 我们以训练集的经验误差不再下降作为模型参数迭代是否终止的标准，根据算法实现的结果来看，随机梯度方法需要在全部的训练数据上跑 1 到 2 次经验误差不再下降，而 IRLS 方法需跑 2 到 3 次。





2.2 SVM

Doing:

b: 我们采用 Linear Kernel, Gaussian Kernel 分别作为核函数，它们分别有 0,1 个额外参数必须提前确定才能求解凸优化问题确定支持向量，另外还有正则化因子需要确定。因此我们针对两种核函数模型，采用 Grid Search 与控制变量相结合的方法寻找使判别误差最小的 Hyper-parameters 的取值。

同样的，我们对数据进行适当的归一化后进行训练和预测。

Report:

a:

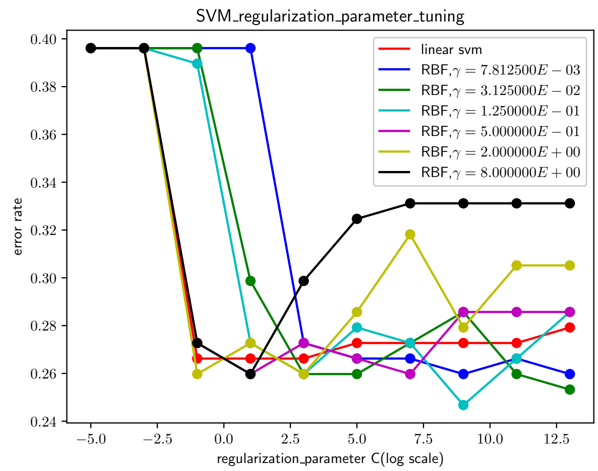
由上图可知，取 RBF 核函数，$C \approx a, \gamma \approx b$ 时可取得最小的预测误差值，约为 c, 该模型的预测误差比其他线性方法均要小。

上机作业使用的 Python 代码：

diabetes.py

```
#x_data loader

import numpy as np
np.seterr(all='raise')
```




```python
import code
class Diabetes_Binary_Classifier:
    def __init__(self,provided_method='lr'):#default method is logistic regression with no regularization
        self.x_data=[]
        self.y_data=[]
        self.w=[]
        self.training_dataset_index=[]
        self.test_dataset_index=[]
        self.method=provided_method
    def _parse_line(self,line_string):
        _Ls=line_string.split(' ')#_Ls[0]=\pm 1,Ls[1]=feature_vector
        assert(len(_Ls)==2)
        float_feature_vector=[float(float_string.split(':')[1]) for float_string in _Ls[1].split(' ')]
        #code.interact(local=locals())
        #append float_feature_vector to numpy x_data array
        self.x_data.append(float_feature_vector)
        self.y_data.append(int(_Ls[0])/2+0.5)
    def _logistic_function(self,x):
        try:
            lf=1/(1+np.exp(-np.dot(self.w,x)))
        except FloatingPointError:
            code.interact(local=locals())
        return lf
        #self.w and x are of narray type
    def load(self,file_name):
        for line_string in open(file_name).read().split('\n'):
            self._parse_line(line_string)
        self.feature_dim=len(self.x_data[0])
```

```
        #output statistics to console
        print("parsed: %s,x_data count:%d,feature_dim:%d"%(file_name,len(self.x_data),self.feature_dim))
        self.x_data=np.array(self.x_data)
        #normalization of each column,[0,1]
        for i in range(self.feature_dim):
            min_tmp=min(self.x_data[:,i])
            max_tmp=max(self.x_data[:,i])
            self.x_data[:,i]=(self.x_data[:,i]-min_tmp)/(max_tmp-min_tmp)
    def training_test_split(self,percertage=0.8):
        #split self.(x,y)data into training and test dataset
        #select 80% len(self.x_data) as training data
        self.training_dataset_index=set(np.random.choice(len(self.x_data),int(percertage*len(self.x_dat
        self.test_dataset_index=[i for i in range(len(self.x_data)) if i not in self.training_dataset_index]
        #the remaining data is left as test dataset

    def cross_validation_setup(self):
        #split the total data into five amounts
        self.five_cv=[[],[],[],[],[]]
        choice_left=list(range(len(self.x_data)))
        for i in range(5):
            selected_num=int(0.2*len(self.x_data))
            if(i==4):
                selected_num=len(choice_left)
            self.five_cv[i]=set(np.random.choice(choice_left,selected_num,replace=False))
            choice_left=list(set(choice_left).difference(self.five_cv[i]))


    def logistic_regression(self,turning_parameter=1):
        #initialize self.w,with dimension equal to len(self.x_data[0])
        self.w=np.zeros(self.feature_dim)
        #process training data one by one, multiple pass

        #we should use measurement of error rate in training_dataset to determine when to stop

        #many parameters are empirical, such as how many turns the training process should take and
        #how to select the turning_parameter, alpha
        iterative_count=0
        last_empirical_error_count=0
        empirical_error_count=len(self.x_data[:,0])
        while(True):
            last_empirical_error_count=empirical_error_count
            empirical_error_count=0
            for i in self.training_dataset_index:
                if(np.dot(self.w,self.x_data[i,:])*(self.y_data[i]*2-1)<0):
                    empirical_error_count+=1
                self.w+=turning_parameter*(self.y_data[i]-self._logistic_function(self.x_data[i,:]))*sel
```

```
            if(last_empirical_error_count<=empirical_error_count):
                break
            self.w/=np.linalg.norm(self.w)
            iterative_count+=1
            #maybe there is no need for the normalization of self.w
            #self.w=self.w/np.linalg.norm(self.w)
            #report the error rate at this pass
        #code.interact(local=locals())
        #print("Logistic Regression, use %d times to converge"%iterative_count)
        return
    def IRLS_cross_validate(self,regularization_parameter):
        #five turn
        error_rate=[]
        for i in range(5):
            #in each turn, assemble training and test set firstly
            self.training_dataset_index=[]
            for j in range(5):
                if(i==j):
                    #assemble test set
                    self.test_dataset_index=self.five_cv[i]
                else:
                    self.training_dataset_index.extend(self.five_cv[j])
            #then run IRLS for the given regularization_parameter
            self.IRLS(regularization_parameter)
            error_rate.append(self._predict()/len(self.test_dataset_index))
        #print('IRLS cross validate for regularization parameter: %f,\n Average error rate for test set: %f'%(regularization_parameter,np.mean(error_rate)))
        return np.mean(error_rate)

    def IRLS(self,regularization_parameter=0):
        #Iterative Reweighted Least Square uses Newton-Raphson iterative method to solve nonlinear function
        self.w=np.zeros(self.feature_dim)
        update_vector=np.zeros(self.feature_dim)
        #calculate gradient and Hessian matrix
        iterative_count=0
        last_empirical_error_count=0
        empirical_error_count=len(self.x_data[:,0])
        while(True):
            last_empirical_error_count=empirical_error_count
            empirical_error_count=0
            gradient=update_vector*regularization_parameter
            Hessian=regularization_parameter*np.identity(self.feature_dim)
            for i in self.training_dataset_index:
                _logistic_temp=self._logistic_function(self.x_data[i,:]);
                gradient+=(self.y_data[i]-_logistic_temp)*self.x_data[i,:]
                Hessian+=_logistic_temp*(_logistic_temp-1)*np.kron(self.x_data[i,:],self.x_data[i,:]).re
                if(np.dot(self.w,self.x_data[i,:])*(self.y_data[i]*2-1)<=0):
```

```
                    empirical_error_count+=1
            update_vector=-np.linalg.solve(Hessian,gradient)
            #if update_vector is very small, stop updating process
            if(last_empirical_error_count<=empirical_error_count):
                break
            self.w+=update_vector
            iterative_count+=1
        #report the iteration times
        #print("IRLS, use %d times to converge"%iterative_count)
        #print("last_empirical_error_count: %f"%last_empirical_error_count)
        #print("empirical_error_count: %f"%empirical_error_count)
        return last_empirical_error_count

    def SVM(self,isLinear=True,gamma=1):
        import svmutil
        #libsvm wrapper
        #first reload the data in svm format
        y_svm=[int(self.y_data[i]*2-1) for i in self.training_dataset_index];
        x_svm=[]
        for i in self.training_dataset_index:
            x_svm_item={}
            for j in range(self.feature_dim):
                x_svm_item[j]=self.x_data[i,j]
            x_svm.append(x_svm_item)
        #generate test data
        y_svm_test=[int(self.y_data[i]*2-1) for i in self.test_dataset_index];
        x_svm_test=[]
        for i in self.test_dataset_index:
            x_svm_test_item={}
            for j in range(self.feature_dim):
                x_svm_test_item[j]=self.x_data[i,j]
            x_svm_test.append(x_svm_test_item)
        error_rate=[]
        c_discrete_log=[(-5+2*i) for i in range(10)]
        #fake for testing
        #return [c_discrete_log,c_discrete_log]
        for i in c_discrete_log:
            if(isLinear):
                svm_train_str='-t 0 -c %f'%(np.exp(i))
            else:
                svm_train_str='-c %f -g %f'%(np.exp(i),gamma)
            libsvm_model = svmutil.svm_train(y_svm,x_svm,svm_train_str)
            _, p_acc, _ = svmutil.svm_predict(y_svm_test, x_svm_test, libsvm_model)
            error_rate.append((100-p_acc[0])/100)
        #error report: (100-p_acc[0])/100
        return [c_discrete_log,error_rate]
```

```
    def text_report(self):
        #generate tabular report
        return
    def graphic_report(self):
        #generate graphic report with matplotlib
        import matplotlib.pyplot as plt
        #(task=='logistic_regression_regularization_parameter_tuning'):
        if(False):
            rp,er=self.logistic_regression_regularization_parameter_tuning()
            plt.plot(rp,er,'ro',rp,er)
            plt.xlabel('regularization_parameter')
            plt.ylabel('error rate')
            plt.title('regularization_parameter_tuning')
            plt.savefig('logistic_regression_regularization_parameter_tuning.eps')
            plt.show()
        #(task=='typical_curve_of_loss_and_accurary'):
        if(False):
            pt,er1,er2=self.typical_curve_of_loss_and_accurary()
            plt.plot(pt,er1,'ro',pt,er2,'b^')
            line_1,=plt.plot(pt,er1,'r-',label='train')
            line_2,=plt.plot(pt,er2,'b-',label='test')
            plt.xlabel('percertage of traing_set')
            plt.ylabel('error rate')
            plt.legend(handles=[line_1,line_2])
            plt.title('typical_curve_of_loss_and_accurary')
            plt.savefig('typical_curve_of_loss_and_accurary.eps')
            plt.show()
        #(task=='SVM hyper-parameter tuning')
        if(True):
            #get linear svm plot data
            self.training_test_split()
            plt.rc('text', usetex=True)
            x_log,er=self.SVM()
            er_Gaussian=[]
            gamma_discrete=[np.power(2.0,1.0*(-7+2*i)) for i in range(6)]
            Guassian_color=['b', 'g', 'c', 'm', 'y', 'k']
            line_bundle=[]
            for i in gamma_discrete:
                _,er_temp=self.SVM(False,i)
                er_Gaussian.append(er_temp)
            plt.plot(x_log,er,'ro')
            linear_line,=plt.plot(x_log,er,'r-',label='linear svm')
            line_bundle.append(linear_line)
            for index,i in enumerate(gamma_discrete):
                plt.plot(x_log,er_Gaussian[index],Guassian_color[index]+'o')
                tmp_line,=plt.plot(x_log,er_Gaussian[index],Guassian_color[index]+'-',label='RBF,$\gamma
                line_bundle.append(tmp_line)
            plt.legend(handles=line_bundle)
            plt.xlabel('regularization\_parameter C(log scale)')
            plt.ylabel('error rate')
```

```
            plt.title('SVM\_regularization\_parameter\_tuning')
            plt.savefig('SVM_regularization_parameter_tuning.eps')
            plt.show()

            code.interact(local=locals())
        return
    def typical_curve_of_loss_and_accurary(self):
        pt=[0.4,0.5,0.6,0.7,0.8,0.9]
        er_1=[]
        er_2=[]
        self.training_dataset_index=set(np.random.choice(len(self.x_data),int(pt[0]*len(self.x_data)),r
        self.test_dataset_index=[i for i in range(len(self.x_data)) if i not in self.training_dataset_index]
        for d in pt:
            er_1.append(self.IRLS()/len(self.training_dataset_index))
            er_2.append(self._predict()/len(self.test_dataset_index))
            #add 10% to the training dataset index
            self.training_dataset_index=set(np.random.choice(self.test_dataset_index,int(0.1*len(self.x
            self.test_dataset_index=[i for i in range(len(self.x_data)) if i not in self.training_dataset_index]
            # we can not random split training and test set in each iteration!
            #self.training_test_split(d)
        return [pt,er_1,er_2]
    def logistic_regression_regularization_parameter_tuning(self):
        print("\n********* Logistic Regression, regularization_parameter tuning process:*********\n")
        self.cross_validation_setup()
        rp=[np.log(i+1) for i in range(10)]
        er=[]
        for i in rp:
            er.append(self.IRLS_cross_validate(i));
        return [rp,er]

    def _predict(self):#use the trained "w" to predict on test dataset
        empirical_error_count=0
        for i in self.test_dataset_index:
            if(np.dot(self.w,self.x_data[i,:])*(self.y_data[i]*2-1)<0):
                empirical_error_count+=1
        return empirical_error_count

    def predict(self):
        #first report the behaviour of algorithm on training dataset
        empirical_error_count=0
        for i in self.training_dataset_index:
            if(np.dot(self.w,self.x_data[i,:])*(self.y_data[i]*2-1)<0):
                empirical_error_count+=1
        print("On Training data,error rate: %f"%(empirical_error_count/len(self.test_dataset_index)))
        empirical_error_count=self._predict()
```

```python
        #report the statitical result to stdout
        print("\n predict_result:\n \ttest_data_count: %s,\
            \n\t empirical_error count:%d,\
            \n\t empirical_error rate:%f"%(len(self.test_dataset_index),empirical_error_count,empirical_error_count/len(
    def _error_report(self,method,turns=10):
        error_vector=[]
        print("\n********* Method: %s:*********\n"%method)
        for i in range(turns):
            self.training_test_split()
            if(method=='logistic_regression'):
                self.logistic_regression()
            elif(method=='IRLS'):
                self.IRLS()
            error_vector.append(self._predict())
        error_rate_mean=np.mean(error_vector)/len(self.test_dataset_index)
        error_rate_var=np.var(error_vector)/(len(self.test_dataset_index)**2)
        print("\t error_rate_mean:%f,error_rate_std_var:%f"%(error_rate_mean,np.sqrt(error_rate_var)))
# use the following code to debug
#
if __name__ == "__main__":
    diabetes_binary_classifier_instance=Diabetes_Binary_Classifier()
    diabetes_binary_classifier_instance.load('diabetes.txt')
    iteration_time=100
    print("\n********* Method: Logistic Regression:*********\n")
    er=[]
    for i in range(iteration_time):
        diabetes_binary_classifier_instance.training_test_split()
        diabetes_binary_classifier_instance.logistic_regression()
        er.append(diabetes_binary_classifier_instance._predict()/len(diabetes_binary_classifier_instanc
    print("err mean= %f, err std var= %f"%(np.mean(er),np.sqrt(np.var(er))))

    print("\n********* Method: Iterative Reweighted Least Square:*********\n")
    er=[]
    for i in range(iteration_time):
        diabetes_binary_classifier_instance.training_test_split()
        diabetes_binary_classifier_instance.IRLS()
        er.append(diabetes_binary_classifier_instance._predict()/len(diabetes_binary_classifier_instanc
    print("err mean= %f, err std var= %f"%(np.mean(er),np.sqrt(np.var(er))))


    diabetes_binary_classifier_instance.graphic_report()
```

(30)