最优化与运筹学实验报告

林聪　　16339026
2016 级信息与计算科学
linc7@mail2.sysu.edu.cn

任务一

考虑线性测量$b = Ax + e$，其中b为50维的测量值，A为50×100维的测量矩阵，x为 100 维的未知稀疏向量且稀疏度为5，e为50维的测量噪声。从b与A中恢复x的一范数规范化最小二乘模型如下：

$$\min_x \|Ax - b\|_2^2 + \frac{p}{2}\|x\|_1$$

其中p为非负的正则化参数。请设计下述算法求解该问题：

1、 邻近点梯度下降法；

2、 交替方向乘子法；

3、 次梯度法；

在实验中，设x的真值中的非零元素服从均值为0方差为1的高斯分布，A中的元素服从均值为0方差为1的高斯分布，e中的元素服从均值为0方差为0.1的高斯分布。对于每种算法，请给出每步计算结果与真值的距离以及每步计算结果与最优解的距离。此外，请讨论正则化参数p对计算结果的影响；

1　问题描述

任务本身已将问题描述清楚，请移步下一节.

2　算法设计

a)　邻近点梯度下降法

对于优化问题$\min_x s(x) + r(x)$，其中$s(x)$为光滑函数，$r(x)$为非光滑函数，邻近点梯度下降法的求解过程为：

$$\begin{cases} x^{k+1/2} = x^k - \alpha \nabla s(x^k) \\ x^{k+1} = \arg\min_x r(x) + \frac{1}{2\alpha}\left\|x - x^{k+1/2}\right\|^2 \end{cases},$$

于是算法在该任务中的求解过程为

$$\begin{cases} x^{k+1/2} = x^k - 2\alpha A^T(Ax - b) \\ x^{k+1} = \arg\min_x \frac{p}{2}\|x\|_1 + \frac{1}{2\alpha}\left\|x - x^{k+1/2}\right\|^2 \end{cases},$$

下面推导邻近点算子$\arg\min_x \frac{p}{2}\|x\|_1+\frac{1}{2\alpha}\left\|x-x^{k+1/2}\right\|^2$：

由于各分量独立，故可以对各分量分别求邻近点算子：

$$x_i^{k+1}=\arg\min_{x_i}\frac{p}{2}|x_i|+\frac{1}{2\alpha}\left(x_i-x_i^{k+1/2}\right)^2,\quad \forall i$$

求次梯度并令次梯度等于 0 得三种情况：

$$\begin{cases} x_i^{k+1}<0, & \frac{1}{\alpha}\left(x_i^{k+1}-x_i^{k+1/2}\right)-\frac{p}{2}=0 \\ x_i^{k+1}>0, & \frac{1}{\alpha}\left(x_i^{k+1}-x_i^{k+1/2}\right)+\frac{p}{2}=0 \\ x_i^{k+1}=0, & \frac{1}{\alpha}\left(x_i^{k+1}-x_i^{k+1/2}\right)\in\left[-\frac{p}{2},\frac{p}{2}\right] \end{cases},\quad \forall i$$

解得邻近点算子为

$$x_i^{k+1}=\begin{cases} x_i^{k+1/2}+\frac{\alpha p}{2}, & x_i^{k+1/2}<-\frac{\alpha p}{2} \\ x_i^{k+1/2}-\frac{\alpha p}{2}, & x_i^{k+1/2}>\frac{\alpha p}{2} \\ 0, & elsewhere \end{cases},\quad \forall i$$

b) **交替方向乘子法**

对于优化问题$\min_{x,y} f_1(x)+f_2(y), s.t.\ Ax+By=d$, 其中$f_1(x)$和$f_2(y)$为凸函数，交替方向乘子法的求解过程为：

$$\begin{cases} x^{k+1}=\arg\min_x L_c(x,y^k,v^k) \\ y^{k+1}=\arg\min_y L_c(x^{k+1},y,v^k) \\ v^{k+1}=v^k+c(Ax^{k+1}+By^{k+1}), \end{cases}$$

其中$L_c(x,y,v)$为增广拉格朗日函数.

对原任务进行改写可得到等价任务$\min_{x,z}\|Ax-b\|_2^2+\frac{p}{2}\|z\|_1, s.t.\ z-x=0$。于是对应的交替方向乘子法的求解过程为：

$$\begin{cases} x^{k+1}=\arg\left\{\frac{\partial L_c(x,z^k,v^k)}{\partial x}=0\right\}=(2A^TA+cI)^{-1}(2A^Tb+cz^k+v^k), \\ z^{k+1}=\arg\min_z \frac{p}{2}\|z\|_1+\frac{c}{2}\left\|z-x^{k+1}\right\|^2+\langle v^k,z\rangle \\ v^{k+1}=v^k+c(z^{k+1}-x^{k+1}) \end{cases},$$

其中z^{k+1}可以通过邻近点算子显式求解，与 a)中同理可得

$$z_i^{k+1}=\begin{cases} x_i^{k+1}-\frac{2v_i^k-p}{2c}, & x_i^{k+1}<\frac{2v_i^k-p}{2c} \\ x_i^{k+1}-\frac{2v_i^k+p}{2c}, & x_i^{k+1}>\frac{2v_i^k+p}{2c} \\ 0, & elsewhere \end{cases},\quad \forall i$$

c) **次梯度法**

对原任务$\min_x\|Ax-b\|_2^2+\frac{p}{2}\|x\|_1$，记$f(x)=\|Ax-b\|_2^2$，$g(x)=\frac{p}{2}\|x\|_1$，则次梯度法的求解过程为

$$x^{k+1} = x^k - \alpha_k \left(\partial f(x^k) + \partial g(x^k) \right),$$

其中由于$f(x)$可微，得$\partial f(x^k) = 2A^T(Ax - b)$，而对不可微函数$g(x)$有

$$\partial g(x^k)_i \in \begin{cases} \left\{\frac{p}{2}\right\}, & x_i^k > 0, \\ \left\{-\frac{p}{2}\right\}, & x_i^k < 0, \\ \left(-\frac{p}{2}, \frac{p}{2}\right), & elsewhere. \end{cases} \quad \forall i$$

整个过程中步长α_k的选取方法为$\alpha_k = \frac{\alpha}{k+1}$或$\alpha_k = \frac{\alpha}{\sqrt{k+1}}$.

3 数值实验

3.1 实验环境

在 Windonw 10 操作系统下，使用 Python 3.6.4 在 Jupyter Notebook 5.6.0 中编写实验代码，涉及到的 package 有： numpy 1.15.2 以及 matplotlib 3.0.0.

3.2 实验细节

- 目标设定：用 numpy 根据任务要求随机生成真值A, x和e，并由此计算出b.
- 参数初始化：对每一种算法，将参与迭代的变量，如x^k, z^k和v^k, 初始化为 0 向量.
- 迭代次数：对前两种算法，每次测试迭代 1000 次，对次梯度法每次迭代 2000 次。之所以这样设计是为了充分观察迭代过程，防止因结果提升过小而在仍有提升空间的情况下提前结束迭代。实际测试时发现次梯度法收敛较慢，故迭代次数相对较多.
- 次梯度选取：当次梯度取值不唯一时，在取值范围内随机均匀取值.
- 结果记录：记录每次迭代后得到的x^k，在迭代结束后，作出每一次结果在 L1 范数下到真值x的距离的变化曲线，以及到最终结果的 L1 范数距离曲线.
- 超参数设置：实验针对不同的设定目标进行若干次，每次对各算法中的各个超参数进行若干次尝试，以寻找较优的超参数.

3.3 实验结果

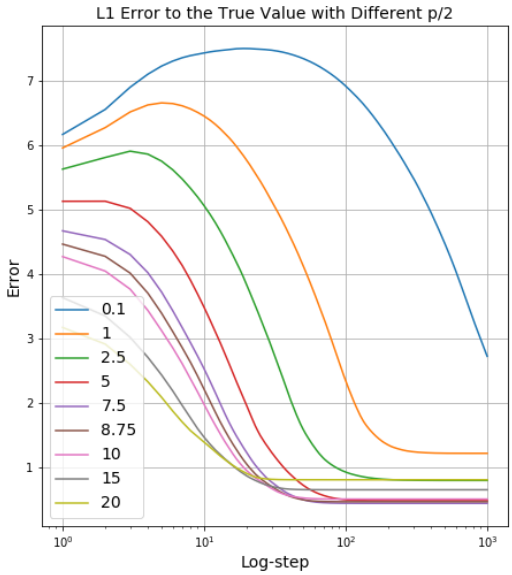
实验总共设定 3 次目标，对于每个目标，下表给出了各个算法的最终结果到真值的最近距离以及相应的超参数，其中次梯度法的步长更新方式为$\alpha_k = \frac{\alpha}{\sqrt{k+1}}$.

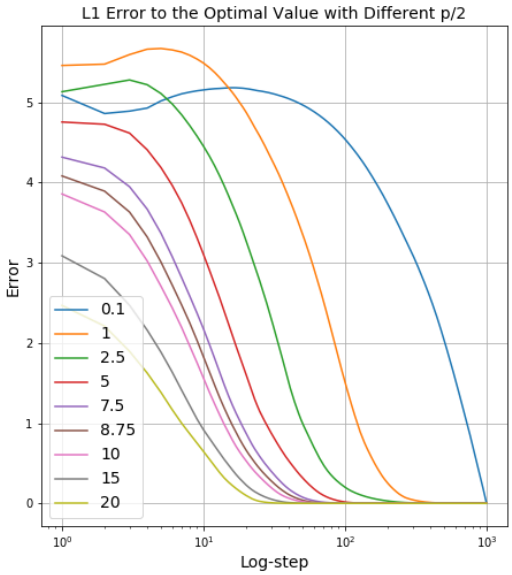
	邻近点梯度下降法			交替方向乘子法			次梯度法		
实验次数	最近距离	对应α	对应$\frac{p}{2}$	最近距离	对应c	对应$\frac{p}{2}$	最近距离	对应α	对应$\frac{p}{2}$
第 1 次	0.3291	0.0025	7.5	0.3291	100	7.5	0.3572	0.0025	7.5
第 2 次	0.5703	0.0025	10	0.5703	100	10	0.5989	0.0025	8.75
第 3 次	0.4442	0.0025	7.5	0.4442	100	7.5	0.4733	0.0025	7.5

其中第三次实验的图像如下：

a) **邻近点梯度下降法**

在$\alpha = 0.0025$的条件下对$\frac{p}{2}$进行测试，结果如下：

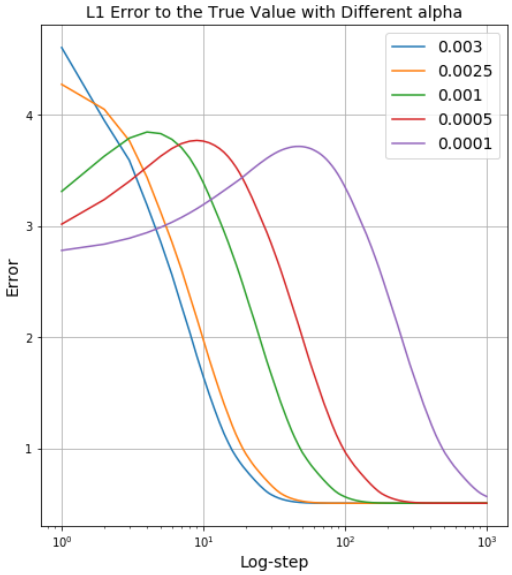


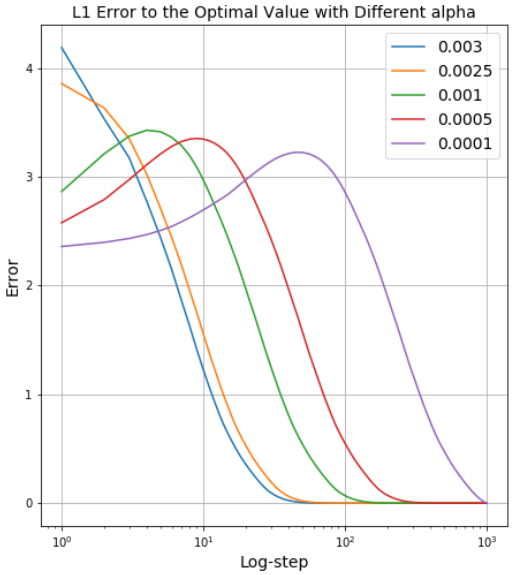


较优的$\frac{p}{2}$及其对应的到真值的距离为

$\frac{p}{2}$	5	7.5	8.75	10
最终距离	0.4864	0.4442	0.4686	0.5100

对在$\frac{p}{2} = 7.5$的条件下对α进行测试，结果如下：

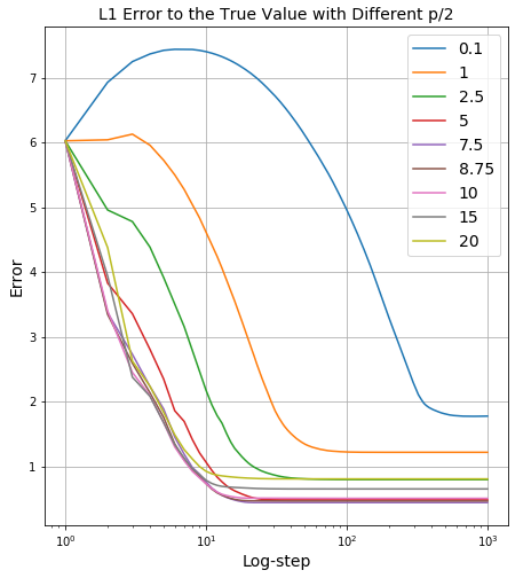


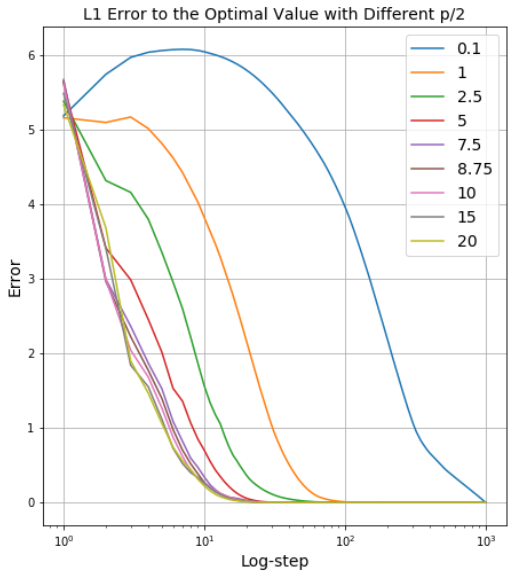


除了$\alpha = 0.0001$的情况外，采用其余步长所获得的最终结果几乎没有区别.

b) **交替方向乘子法**

在$c = 100$的条件下对$\frac{p}{2}$进行测试，结果如下：

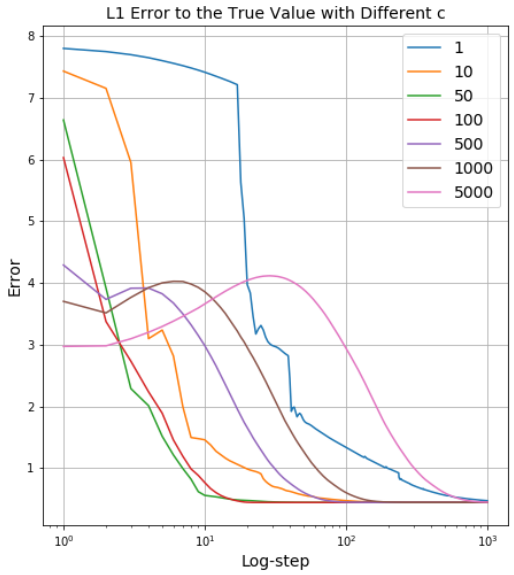


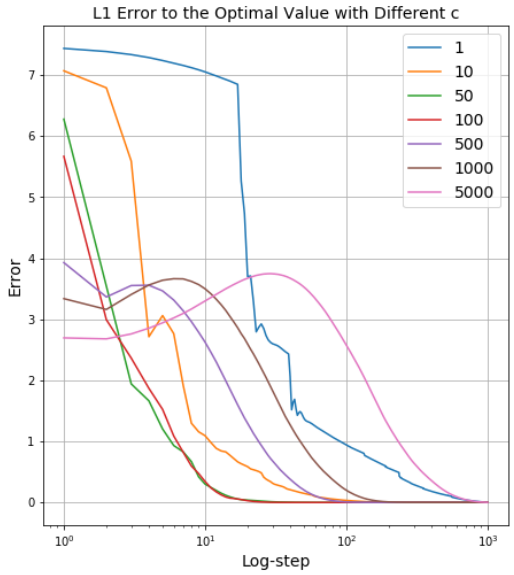


较优的$\frac{p}{2}$及其对应的到真值的距离为

$\frac{p}{2}$	5	7.5	8.75	10
最终距离	0.4864	0.4442	0.4686	0.5100

对在$\frac{p}{2} = 7.5$的条件下对c进行测试，结果如下：

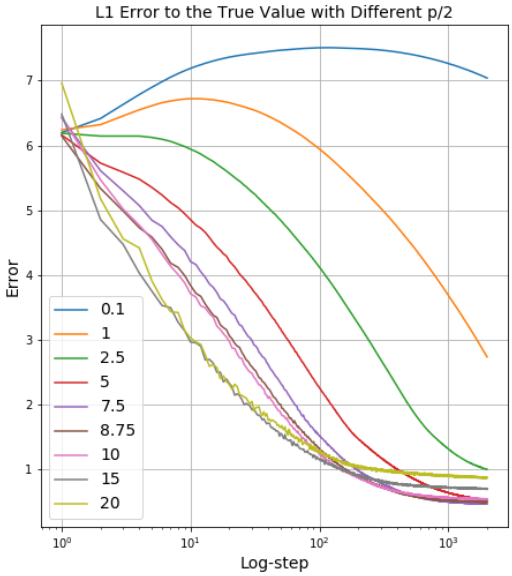


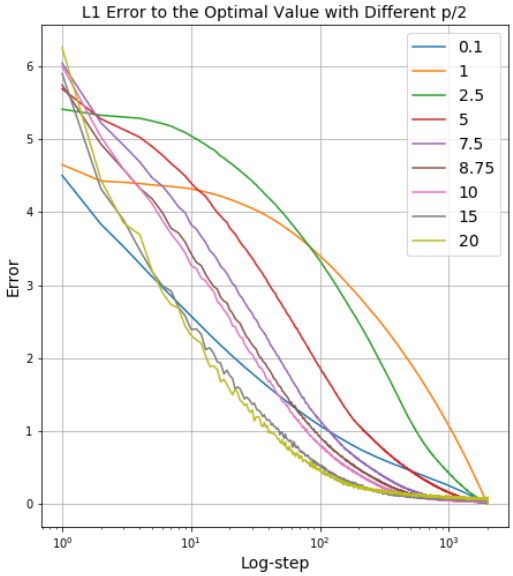


不同的c产生的最终结果几乎没有区别.

c) 次梯度法

在$\alpha = 0.0025$的条件下对$\frac{p}{2}$进行测试，结果如下：

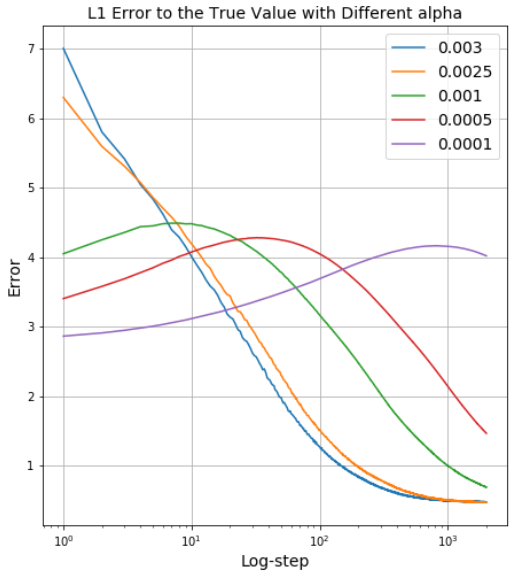


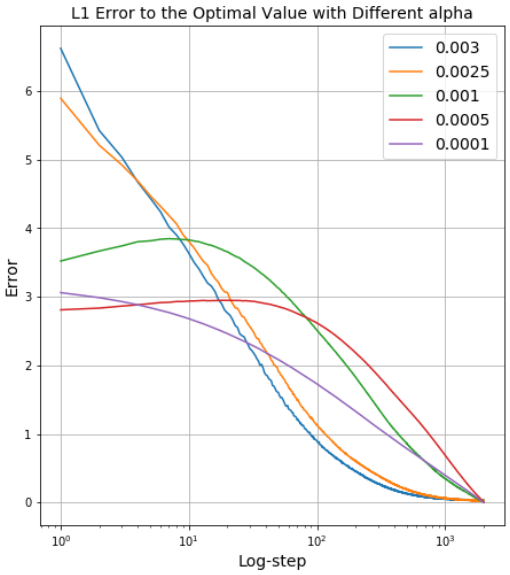


较优的$\frac{p}{2}$及其对应的到真值的距离为

$\frac{p}{2}$	5	7.5	8.75	10
最终距离	0.5334	0.4733	0.5015	0.5478

对在$\frac{p}{2}=7.5$的条件下对α进行测试，结果如下：





不同的α及其对应的到真值的距离为

α	0.003	0.0025	0.001	0.0005	0.0001
最终距离	0.4803	0.4733	0.6916	1.4713	4.0177

4 结果分析

三种方法所得到的结果相近，其中邻近点梯度下降法和交替方向乘子法所得的结果一致，次梯度法所得结果略大。各自的最优结果与真值相减并取绝对值后，得到的向量中的元素几乎全为 0，只有不到 10 个在10^{-2}量级的非零元素，这说明对真值的恢复效果很好。此外实

验发现，利用三种算法求解该任务时，$\frac{p}{2}$的最优值总是在 5 到 10 之间，大部分情况下取 7.5 就能得到最好的结果。对三种算法的具体分析如下：

a) **邻近点梯度下降法**

- 在同一合适的α下，越小的$\frac{p}{2}$会导致结果越晚收敛并且收敛越慢，最终距离也较大，并且注意到$\frac{p}{2}=0.1$时到真值的距离曲线和到最终结果的距离曲线形状不同，说明参数已经不是朝着真值收敛的了。越大的$\frac{p}{2}$越快收敛，在 100 次迭代内可以平稳，收敛幅度也越小，但最终到真值的距离可能会不降反升。一般来说$\frac{p}{2}$在 5 到 10 之间取值可以获得最好的结果.
- 在同一合适的$\frac{p}{2}$下，越小的α越晚收敛，并且在迭代初期会经历越长的发散阶段，但经过足够的迭代次数后所得的结果没有差别。较大的α可以避免迭代初期的发散，直接进入收敛阶段，收敛速度也更快，在 100 次迭代内可以平稳。但一般来说当α超过 0.003 时，迭代过程很有可能失控，产生巨大量级的结果。综合收敛速度和收敛过程，α取 0.0025 总是非常好的选择.

b) **交替方向乘子法**

- 在同一合适的c下，越小的$\frac{p}{2}$会导致结果越晚收敛并且收敛越慢，最终结果也较大，但参数是朝真值方向收敛的。越大的$\frac{p}{2}$越快收敛，在 100 次迭代内可以平稳，收敛幅度也越小，但最终到真值的距离可能会不降反升。一般来说$\frac{p}{2}$在 5 到 10 之间取值可以获得最好的结果.
- 在同一合适的$\frac{p}{2}$下，越小的c越晚收敛，并且收敛过程越不平滑，但经过足够的迭代次数后所得的结果没有太大差别。较大的c可以更早进入收敛阶段，收敛速度也更快，收敛过程也越平滑，在 100 次迭代内可以平稳。但太大的c，例如 500 以上，会导致迭代初期出现发散情况，也越晚进入收敛阶段。综合收敛速度和收敛过程，c取 100 总是非常好的选择.

c) **次梯度法**

- 实验发现对于该任务，步长采用$\alpha_k=\frac{\alpha}{k+1}$效果普遍不好，故采用$\alpha_k=\frac{\alpha}{\sqrt{k+1}}$.
- 在同一合适的α下，越小的$\frac{p}{2}$会导致结果越晚收敛并且收敛越慢，最终距离也较大，并且注意到$\frac{p}{2}=0.1$时到真值的距离曲线和到最终结果的距离曲线形状不同，说明参数已经不是朝着真值收敛的了。越大的$\frac{p}{2}$越快收敛，能够直接进入收敛阶段，在

100 次迭代内可以平稳，收敛幅度也越小，但收敛前不平滑性比较明显，太大的$\frac{p}{2}$可以使得最终到真值的距离不降反升。一般来说$\frac{p}{2}$在 5 到 10 之间取值可以获得最好的结果.

- 在同一合适的$\frac{p}{2}$下，越小的α越晚收敛，并且在迭代初期会经历越长的发散阶段。较大的α可以避免迭代初期的发散，直接进入收敛阶段，收敛速度也更快，在 1000 次迭代内可以平稳。但一般来说当α超过 0.003 时，迭代过程很有可能失控，产生巨大量级的结果。综合收敛速度和收敛过程，α取 0.0025 总是非常好的选择.

任务二

请设计下述算法，求解 MNIST 数据集上的 Logistic Regression 问题：

1、梯度下降法；

2、随机梯度法；

对于每种算法,请给出每步计算结果与最优解的距离以及每步计算结果在测试集上所对应的分类精度。此外，请讨论随机梯度法中 Mini Batch 大小对计算结果的影响；

1 问题描述

MNIST 数据集包含 70000 张手写数字图像样本，其中 60000 张用作训练集，10000 张用作测试集。记任一样本为$(x^{(i)}, y^{(i)})$，其中$x^{(i)}$为 784 维的图像像素值向量，$y^{(i)} \in \{0,1,\dots,9\}$为图像的标签。任务要求 MNIST 训练集训练一个 Logistic 分类模型，使其在 MNIST 测试集上的分类正确率，即根据$x^{(i)}$给出正确的$y^{(i)}$的能力，尽可能地高.

具体而言，由于标签的可能取值超过两个，所以实际上要训练的是 Logistic 分类模型的推广——Softmax 分类模型，其原理如下：

给出一个样本$x^{(i)}$，参数为θ, σ的 Softmax 分类模型返回一个分类概率向量$h_{\theta,\sigma}(x^{(i)})$：

$$h_{\theta,\sigma}(x^{(i)}) = \begin{pmatrix} P\{y^{(i)}=0|x^{(i)};\theta,\sigma\} \\ P\{y^{(i)}=1|x^{(i)};\theta,\sigma\} \\ \vdots \\ P\{y^{(i)}=9|x^{(i)};\theta,\sigma\} \end{pmatrix} = \frac{1}{\sum_{j=0}^{9} e^{\theta_j^T x^{(i)}+\sigma_j}} \begin{pmatrix} e^{\theta_0^T x^{(i)}+\sigma_0} \\ e^{\theta_1^T x^{(i)}+\sigma_1} \\ \vdots \\ e^{\theta_9^T x^{(i)}+\sigma_9} \end{pmatrix},$$

其中

$$\theta = \begin{pmatrix} \theta_0^T \\ \theta_1^T \\ \vdots \\ \theta_9^T \end{pmatrix} \in R^{10\times 784}, \qquad \sigma = \begin{pmatrix} \sigma_0 \\ \sigma_1 \\ \vdots \\ \sigma_9 \end{pmatrix} \in R^{10\times 1}.$$

Softmax 分类模型的训练过程即为，给出m个样本$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$，通过不断更新$\theta, \sigma$，使得损失函数

$$\mathrm{J}(\theta,\sigma) = \frac{-1}{m}\left[\sum_{i=1}^{m}\sum_{j=0}^{9} \mathbf{1}\{y^{(i)}=j\} \cdot \ln\left[h_{\theta,\sigma}(x^{(i)})_j\right]\right]$$

的值尽可能达到最小，其中 **1**{·}为示性函数.

2 算法设计

根据链式法则不难求得θ,σ对$\mathrm{J}(\theta,\sigma)$的梯度分别为

$$\nabla_{\theta_k}\mathrm{J}(\theta,\sigma)=\frac{-1}{m}\sum_{i=1}^{m}\left[\mathbf{1}\{y^{(i)}=k\}-h_{\theta,\sigma}\left(x^{(i)}\right)_k\right]\cdot x^{(i)},\ \ k=0,1,\dots,9.$$

$$\nabla_{\sigma_k}\mathrm{J}(\theta,\sigma)=\frac{-1}{m}\sum_{i=1}^{m}\left[\mathbf{1}\{y^{(i)}=k\}-h_{\theta,\sigma}\left(x^{(i)}\right)_k\right],\ \ k=0,1,\dots,9.$$

故可以设计如下基于梯度的算法来更新θ,σ从而训练 Softmax 模型：

a) 梯度下降法

Randomly initialize θ,σ.

for number of training epochs **do**

- Input 60000 training samples $\left(x^{(1)},y^{(1)}\right),\left(x^{(2)},y^{(2)}\right),\dots,\left(x^{(60000)},y^{(60000)}\right)$.
- Output $h_{\theta,\sigma}\left(x^{(1)}\right),h_{\theta,\sigma}\left(x^{(2)}\right),\dots,h_{\theta,\sigma}\left(x^{(60000)}\right)$.
- Update θ,σ by their gradient:

$$\theta_k=\theta_k-\alpha\nabla_{\theta_k}\mathrm{J}(\theta,\sigma),\ \ k=0,1,\dots,9.$$
$$\sigma_k=\sigma_k-\alpha\nabla_{\sigma_k}\mathrm{J}(\theta,\sigma),\ \ k=0,1,\dots,9.$$

end for

其中用于更新θ,σ的步长参数α是一个预先调试好的超参数.

b) 随机梯度法

Randomly initialize θ,σ.

in epoch i out of number of training epochs **do**

- Input batch of m random training samples $\left(x^{(1)},y^{(1)}\right),\left(x^{(2)},y^{(2)}\right),\dots,\left(x^{(m)},y^{(m)}\right)$.
- Output $h_{\theta,\sigma}\left(x^{(1)}\right),h_{\theta,\sigma}\left(x^{(2)}\right),\dots,h_{\theta,\sigma}\left(x^{(m)}\right)$.
- Update θ,σ by their gradient:

$$\theta_k=\theta_k-\alpha_i\nabla_{\theta_k}\mathrm{J}(\theta,\sigma),\ \ k=0,1,\dots,9,$$
$$\sigma_k=\sigma_k-\alpha_i\nabla_{\sigma_k}\mathrm{J}(\theta,\sigma),\ \ k=0,1,\dots,9,$$
$$\text{where }\alpha_i=\frac{\alpha}{i+1}\text{ or }\alpha_i=\frac{\alpha}{\sqrt{i+1}}\text{ throughout the training.}$$

end for

其中批大小m以及用于更新θ,σ的步长系数α是预先规定好的超参数.

3 数值实验

3.1 实验环境

在 Windonw 10 操作系统下，使用 Python 3.6.4 在 Jupyter Notebook 5.6.0 中编写实验代码，涉及到的 package 有：torch 0.4.1，torchvision 0.2.1，numpy 1.15.2 以及 matplotlib 3.0.0.

3.2 实现细节

- GPU 加速：使用了一块 8G 的 NVIDIA GeForce GTX 965M 显卡加速训练.
- 数据归一化：将 MNIST 数据集的图像像素取值从$[0,255]$内的整数归一化到$[0,1]$.

- 参数初始化：θ, σ的初始值服从分布$0.1 \cdot Normal(0,1)$.
- 基于 Pytorch 框架：代码中的 GPU 加速、矩阵操作、后向传播等细节是使用 torch 实现的.

更多细节请查看 ipynb 源代码文件或其 pdf 文件.

3.3 实验过程

a) 梯度下降法

在步长参数α为 1、0.5、0.1、0.05、0.01、0.005、0.001 的情况下分别用训练集对模型进行 1500 轮训练，之所以不使用分类正确率的变化幅度作为停止准则，是因为一方面在变化幅度很小以至于丢失精度时模型仍有很大的提升空间，另一方面算力也允许训练较高的轮数.

每轮更新后只记录得到的参数θ，以及模型在整个测试集上的分类正确率。训练完成后将最终的θ视为最优解，计算先前每轮训练的θ在矩阵的 Frobenius 范数下到最优解的距离.

b) 随机梯度法

先对$\alpha_i = \frac{\alpha}{i+1}$和$\alpha_i = \frac{\alpha}{\sqrt{i+1}}$两种步长更新方式分别调试出合适的$\alpha$，然后在两种更新方式下，分别对批大小$m$为 10、50、100、500、1000、5000、10000 的情况对模型进行 2000 轮训练。每轮更新后只记录得到的参数θ，以及模型在整个测试集上的分类正确率。训练完成后将最终的θ视为最优解，计算先前每轮训练的θ在矩阵的 Frobenius 范数下到最优解的距离.

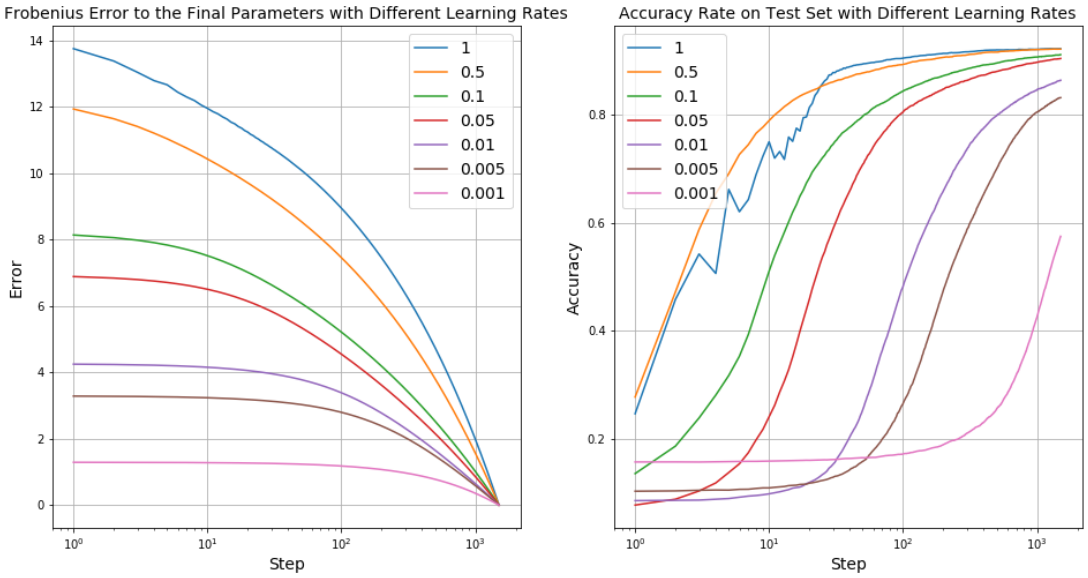
3.4 实验结果

a) 梯度下降法

实验重复了 3 次，所得的分类精度结果如下：

	步长	1	0.5	0.1	0.05	0.01	0.005	0.001
分类精度	第 1 次	92.37%	92.20%	91.27%	90.44%	86.38%	82.90%	56.90%
	第 2 次	92.27%	92.20%	91.17%	90.47%	86.44%	83.20%	57.52%
	第 3 次	92.28%	92.15%	91.13%	90.34%	86.59%	83.01%	59.14%
	平均值	92.31%	92.18%	91.19%	90.42%	86.47%	83.04%	57.85%

三次实验的曲线图相近，其中第 2 次的曲线图如下：



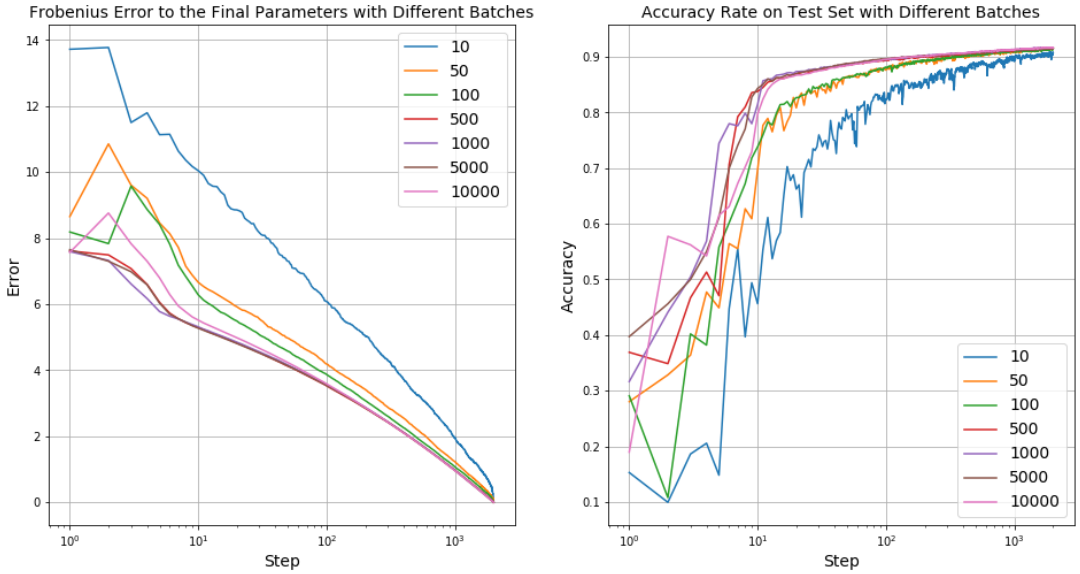
通过更细致的可以发现 0.5 大约是能够使分类正确率平滑上升的最大步长.

b) 随机梯度法

步长$\alpha_i = \frac{\alpha}{\sqrt{i+1}}$时，取$\alpha = 2.5$较为合适。实验重复了 3 次，所得的分类精度结果如下：

	批大小	10	50	100	500	1000	5000	10000
分类精度	第 1 次	90.63%	91.41%	91.45%	91.45%	91.58%	91.52%	91.60%
	第 2 次	90.81%	91.52%	91.40%	91.64%	91.62%	91.56%	91.65%
	第 3 次	90.63%	91.54%	91.37%	91.52%	91.48%	91.35%	91.41%
	平均值	90.69%	91.49%	91.41%	91.54%	91.56%	91.48%	91.55%

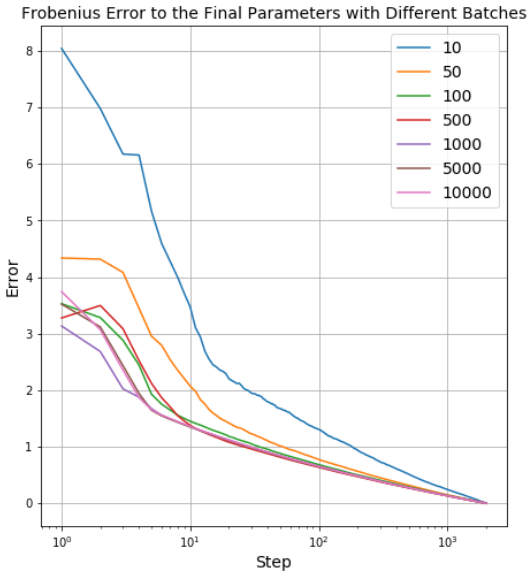
三次实验的曲线图相近，其中第 2 次的曲线图如下：

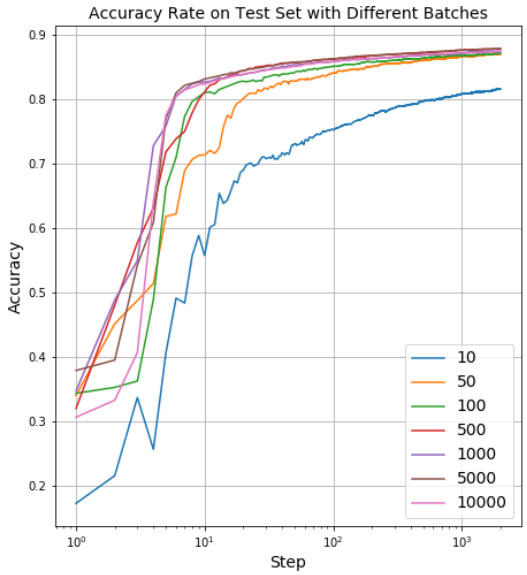


步长$\alpha_i = \frac{\alpha}{i+1}$时，取$\alpha = 2.5$较为合适。实验重复了 3 次，所得的分类精度结果如下：

	批大小	10	50	100	500	1000	5000	10000
分类精度	第 1 次	81.74%	86.18%	86.87%	87.73%	87.65%	87.39%	87.19%
	第 2 次	81.54%	86.91%	87.04%	87.75%	87.40%	87.81%	87.30%
	第 3 次	83.17%	86.09%	86.48%	87.30%	87.82%	87.57%	87.51%
	平均值	82.15%	86.39%	86.80%	87.59%	87.62%	87.59%	87.33%

三次实验的曲线图相近，其中第 2 次的曲线图如下：





4 结果分析

a) **梯度下降法**

- 从最终分类正确率来看，在相同的训练轮数下，步长越大，分类正确率越高。其中步长为 1 时的平均正确率 92.31%相比步长为 0.5 时的平均正确率 92.18%而言，提升不大.
- 从阶段分类正确率来看，步长越小，正确率收敛速度越慢，正确率曲线整体越往右移。但步长过大时，曲线将失去平滑性，正确率在局部甚至会下降。原因可能是过大的步长使得参数下降过度了，从而让参数优化的方向改变，体现在测试集上便是正确率的非单调性.
- 从与最优解的距离来看，根据曲线形态相同，可知不同步长的收敛阶数相同，但收敛幅度不同。步长越大，初始距离越大，收敛幅度也越大。另外，同一训练阶段下不同步长对应的距离大体与步长有某种比例关系，即步长越相近，则距离也越相近.

b) **随机梯度法**

- 从最终分类正确率来看，在相同的训练轮数下，较大的批大小可以产生较高的分类正确率，但是它们的差别并不显著.
- 从阶段分类正确率来看，较大的批大小能够产生较快的正确率收敛速度，并且曲线更早摆脱震荡，这是因为样本越多则模型更容易学习到数据集的信息.
- 从与最优解的距离来看，根据曲线形态相似，不同批大小的收敛阶数相近，但收敛幅度不同。批大小越小，初始距离越大，收敛幅度也越大。对较大的批大小而言，收敛幅度几乎相等，也更快趋于平缓.
- 采用$\alpha_i = \frac{\alpha}{\sqrt{i+1}}$比采用$\alpha_i = \frac{\alpha}{i+1}$所得到的分类正确率普遍较高，这或许是因为前者始终比后者大，可以提供更快的收敛速度.