神经网络和深度学习（类脑）Week 2 作业

姓名: 王冰怡

2021 年 10 月 10 日

题目 1. PCA 实现作业描述：分两种方法 (直接调用包和自己实现)，实现 PCA 对波士顿房价数据集, 计算主成分 pc1,pc2, 并画图展示。

解答. 左图是用 sklearn 的 PCA 方法计算出的结果。
右图是用 numpy，归一化数据后，计算出协方差矩阵的特征值从而得到的结果。
据我观察并没有明显的差异。

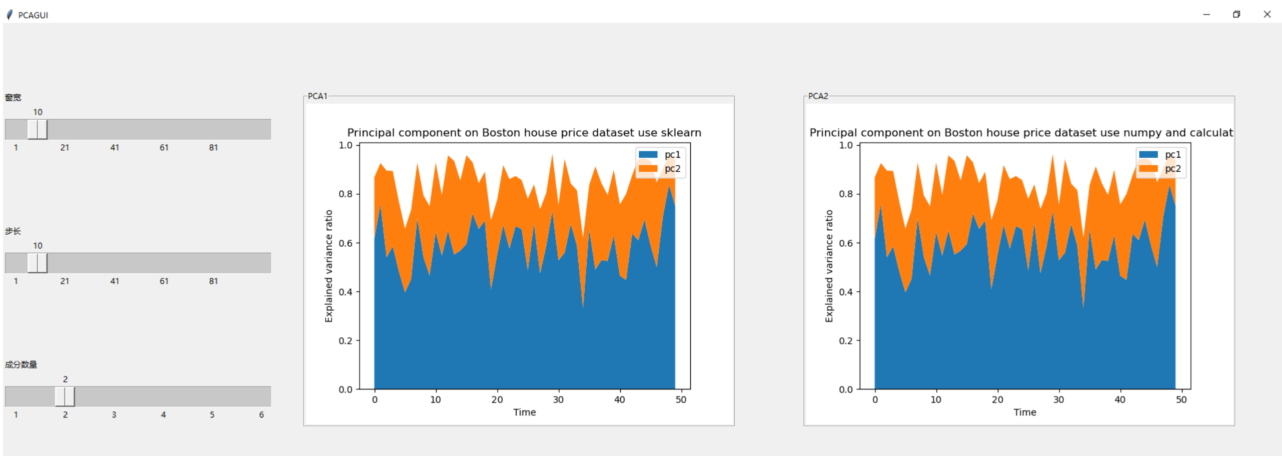


图 1: 利用 GUI 调参数

关于主成分数量对堆积图的影响，可见前两个主成分几乎可以表达 90% 的方差。如果将出成分拉满，可以看到 6 个成分的方差，是如何分布的。

首先分析一下数据，后 6 列的数据分别是
(1). 到波士顿五个中心区域的加权距离；
(2). 辐射性公路的接近指数；
(3). 每 10000 美元的全值财产税率；
(4). 城镇师生比例；
(5). 城镇中黑人的比例。

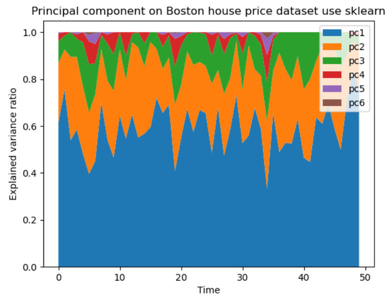
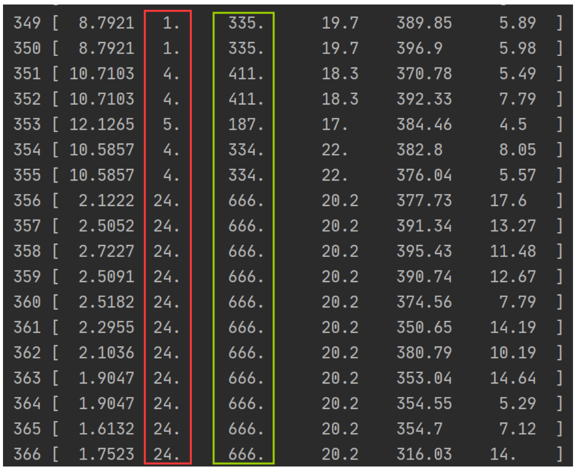


图 2: 6 个成分占据不同比例

(6). 人口中地位低下者的比例。

不同数据的维度变化幅度相差很大，如果不进行标准化，变化量级比较大的数据的方差会掩盖变化量级小的数据。



349	[8.7921	1.	335.	19.7	389.85	5.89]
350	[8.7921	1.	335.	19.7	396.9	5.98]
351	[10.7103	4.	411.	18.3	370.78	5.49]
352	[10.7103	4.	411.	18.3	392.33	7.79]
353	[12.1265	5.	187.	17.	384.46	4.5]
354	[10.5857	4.	334.	22.	382.8	8.05]
355	[10.5857	4.	334.	22.	376.04	5.57]
356	[2.1222	24.	666.	20.2	377.73	17.6]
357	[2.5052	24.	666.	20.2	391.34	13.27]
358	[2.7227	24.	666.	20.2	395.43	11.48]
359	[2.5091	24.	666.	20.2	390.74	12.67]
360	[2.5182	24.	666.	20.2	374.56	7.79]
361	[2.2955	24.	666.	20.2	350.65	14.19]
362	[2.1036	24.	666.	20.2	380.79	10.19]
363	[1.9047	24.	666.	20.2	353.04	14.64]
364	[1.9047	24.	666.	20.2	354.55	5.29]
365	[1.6132	24.	666.	20.2	354.7	7.12]
366	[1.7523	24.	666.	20.2	316.03	14.]

图 3: 截取一部分数据，发现不同属性量级不同

标准化的过程为

$$\mu = \sum_{1}^{n} x_i$$

$$\sigma^2 = \frac{\sum_1^n (x_i - \mu)^2}{n-1}$$

$$X_{norm} = \frac{X - \mu}{\sigma}$$

由于原始数据本身并不是按照某个时间顺序排列起来的，我手动分析了一下数据，感觉数据是按照社区划分的。比方说某个社区的十几条数据在一起，另一个社区的数据在一起。当窗宽 = 步长时，如果相邻两点不变化不大，可能这两个窗都在描述同一个社区的房子信息。如果变化很大，可能当前窗跨越了，或者横跨了多个社区的房价信息。

编程过程中的一些心得：以前在写算法的时候，主要是用 C++，要处理矩阵信息要写各种 for 循环。由于 python 是解释性语言，大量 for 循环效率会很低。但是 numpy 有很多强大的矩阵运算功能，还有切片功能，利用 numpy 强大的矩阵运算，比如乘法，求逆，还有广播，能够发挥出多核 CPU 的运算能力，以及代码比较简洁。比方说求协方差矩阵，设矩阵 X 存放的是行向量，只需要 $\frac{X^T X}{n-1}$ 即可。
但是对人来说，思考量并没有减少，需要考虑清楚数据的维度，究竟是行向量还是列向量，进行运算时维度要匹配，对于很多 numpy 函数的 axis 参数，需要很熟练才行。

题目 2. 求 $f(x, y) = x^3 + y^3 - 3xy$ 的极小值
要求如下：
(1) 使用最速下降和牛顿法两种方法，梯度和 Hessian 矩阵手动求解，直接带入计算。
(2) 可视化求解过程。
(3) 学有余力的同学可以考虑下不同学习率的影响。

解答. 首先分析函数的性质。

$$f(x, y) = x^3 + y^3 - 3xy$$

$$\frac{\partial f(x,y)}{\partial x} = 3x^2 + -3y$$
$$\frac{\partial f(x,y)}{\partial y} = 3y^2 + -3x$$
$$\frac{\partial^2 f(x,y)}{\partial x^2} = 6x$$
$$\frac{\partial^2 f(x,y)}{\partial x \partial y} = -3$$
$$\frac{\partial^2 f(x,y)}{\partial y^2} = 6y$$

令

$$\begin{cases} \frac{\partial f(x,y)}{\partial x} = 3x^2 + -3y = 0 \\ \frac{\partial f(x,y)}{\partial y} = 3y^2 + -3x = 0 \end{cases} \tag{1}$$

得到两个驻点 $(0,0),(1,1)$

由二元函数取极值的充分条件

$$\begin{cases} \frac{\partial^2 f(x,y)}{\partial x^2} = A \\ \frac{\partial^2 f(x,y)}{\partial x \partial y} = B \\ \frac{\partial^2 f(x,y)}{\partial y^2} = C \end{cases} \tag{2}$$

则

$$\Delta = B^2 - AC \begin{cases} < 0 \text{ 是极值} \begin{cases} A < 0 \text{ 是极大值} \\ A > 0 \text{ 是极小值} \end{cases} \\ > 0 \text{ 不是极值} \\ = 0 \text{ 该法失效} \end{cases} \tag{3}$$

(这是高等数学教材的知识，$AC - B^2$ 就是 Hessian 矩阵的行列式值，也可以通过求解 Hessian 行列式值和 A 的大小判断是否是极值点)

牛顿法:

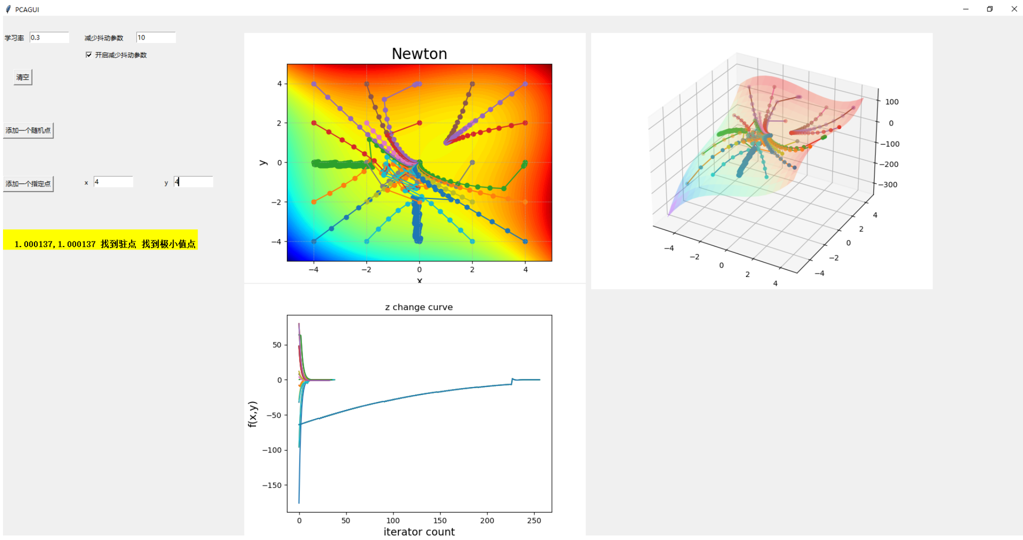


图 4: 随机不同的点利用牛顿法求解驻点

从图片中可以看到，初始化不同位置的店，随着迭代次数的增加，最终都收敛于 $(0,0)$ 或 $(1,1)$

通过函数可以求解出函数的 Hessian 矩阵 $\begin{bmatrix} 6x & -3 \\ 3 & 6y \end{bmatrix}$

若 $xy = \frac{1}{4}$ 则 Hessian 矩阵不可逆。若 $xy \approx \frac{1}{4}$，则 Hessian 矩阵的逆矩阵里的数可能会变得很大，乘以梯度以后是个比较大的向量，因此在图上可以看出，曲线有时会朝某个方向变化很大。

当然有时会一下子跳的很远，为了解决这个问题，一方面 Momentum 可能是个不错的方法。当然也有别的方法。由牛顿法的迭代公式

$$x_k = x_{k-1} - H(x_{k-1})^{-1}\nabla f(x_{k-1})$$

若 $H(x_{k-1})^{-1}\nabla f(x_{k-1})$ 很大，可以在它的基础上除以它的范数，即

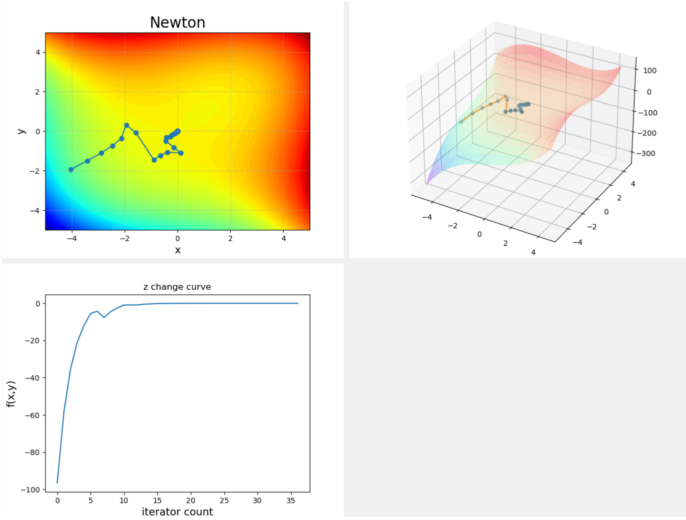


图 5: 收敛的途中不是很稳定

$$
x_k = \begin{cases} x_{k-1} - \alpha H(x_{k-1})^{-1}\nabla f(x_{k-1}), when\|H(x_{k-1})^{-1}\nabla f(x_{k-1})\|_2 < 5 \\ x_{k-1} - \frac{\alpha H(x_{k-1})^{-1}\nabla f(x_{k-1})}{\alpha_2\|H(x_{k-1})^{-1}\nabla f(x_{k-1})\|_2}, when\|H(x_{k-1})^{-1}\nabla f(x_{k-1})\|_2 \geq 5 \end{cases} \tag{4}
$$

其中 α 是学习率，α_2 是减少抖动的参数。

但是这样做有可能导致收敛所需要的迭代次数增加。

当把学习率从 0.1 调到 2 时，会导致牛顿法不收敛。
梯度法:

由于函数不收敛，梯度法会导致某些起始点向地图边界搜索。

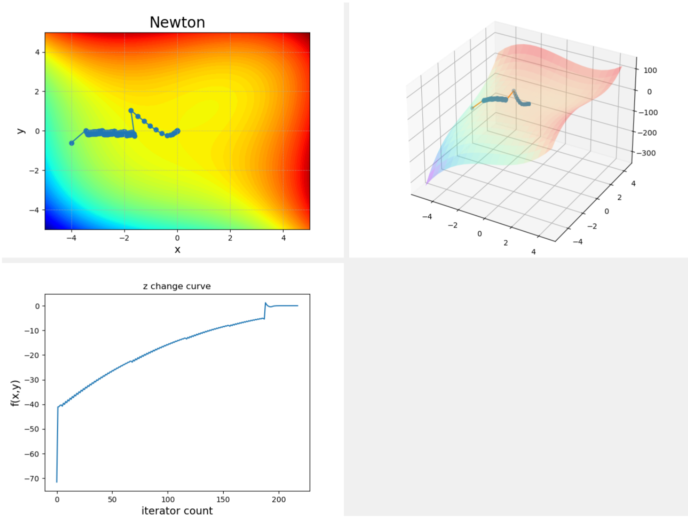


图 6: 收敛所需要的迭代次数增加

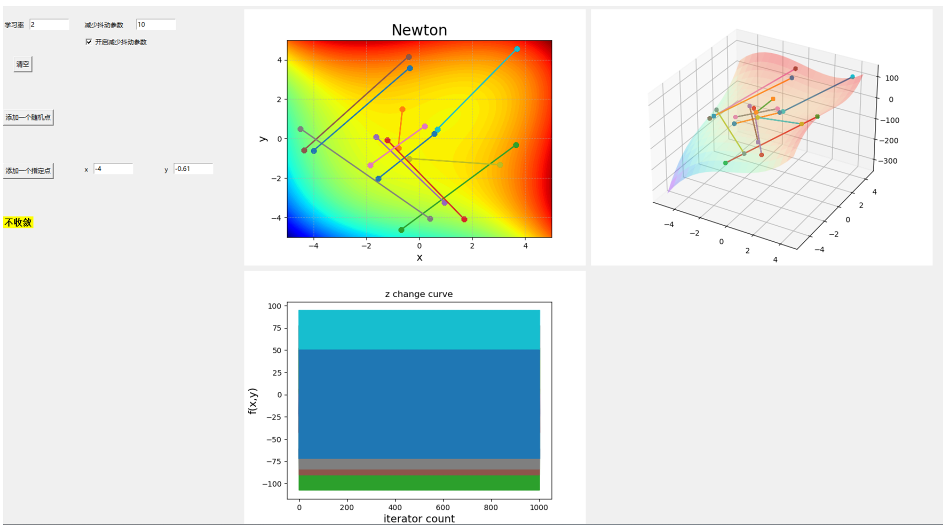


图 7: 学习率过大会导致不收敛

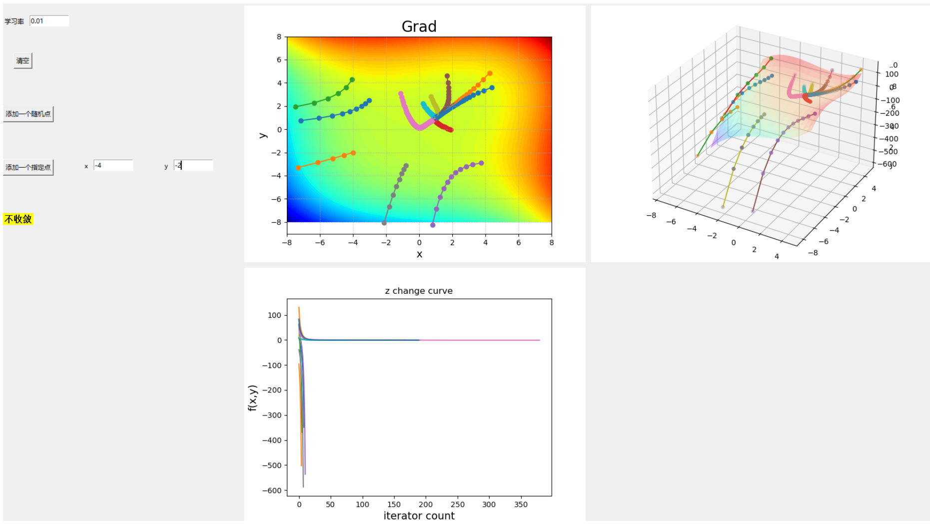


图 8: 梯度法只有部分点能收敛至极小值点，没有点收敛至鞍点