Homework 1

张思源　　*21110850018*

October 14, 2021

1 Ex1

首先, 调用 sklearn.optimize 中的 PCA 方法, 得到结果如下:

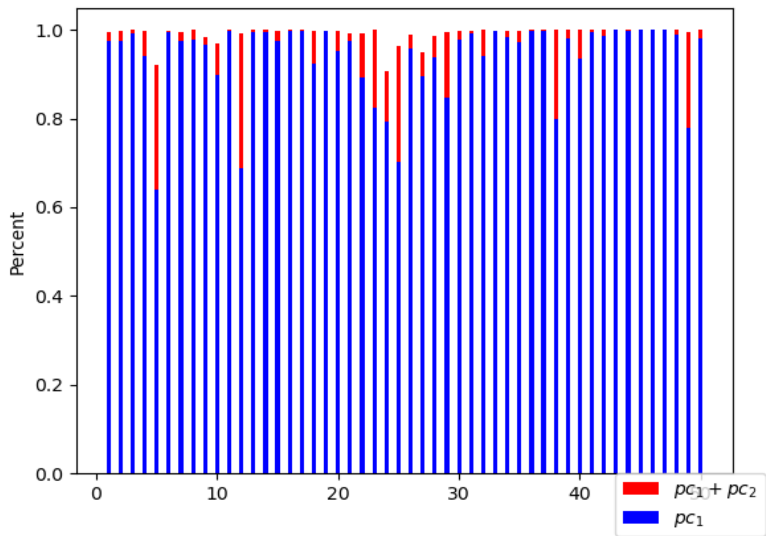

Figure 1: PCA result with 2 principal components by sklearn

然后, 根据最大化方差的观点, 若设 $\mathbf{x}$ 是 m 维随机变量,Σ 是 $\mathbf{x}$ 的协方差矩阵,Σ 的特征值从大到小依次是 $\lambda_1, \lambda_2, \ldots, \lambda_m \geq 0$, 其对应的特征向量依次是 $\alpha_1, \alpha_2, \ldots, \alpha_m$, 则 $\mathbf{x}$ 对应的第 k 主成分是 $y_k = \alpha_k \dot{x}$, 其主成分占比为 $\frac{\lambda_k}{\sum_{i=1}^{m} \lambda_i}$.

进而编写了函数 $mypca(data, n = 2, m = 6, t = 10)$. 其中,$data = matrix$ 表示要做主成分的数据集, 其中这个矩阵的行表示每个 feature 的不同观测, 列表示不同的 feature.n 为保留的主成分数,m 为随机变量的维数, 事实上默认通过输入矩阵计算得出,t 为时间窗的长度. 通过该方法得到的结果如下:

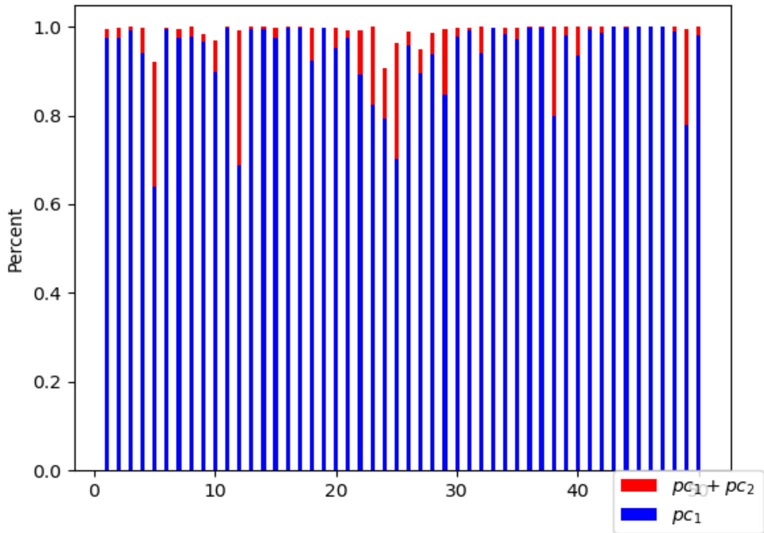

Figure 2: PCA result with 2 principal components by myself

通过对比可以发现两种方法求得的 PCA 第一主成分和第二主成分的占比还是较为接近的.

2 Ex2

首先考虑较为简单的 Newton 法.Newton 法即每次前进的方向为 $-H^{-1}\nabla \mathbf{f}$, 其中 H 为目标函数的 Hessian 矩阵,$\nabla \mathbf{f}$ 为目标函数的梯度.Newton 法有着较快的收敛速度, 但是在实验中也出现了较多的问题:

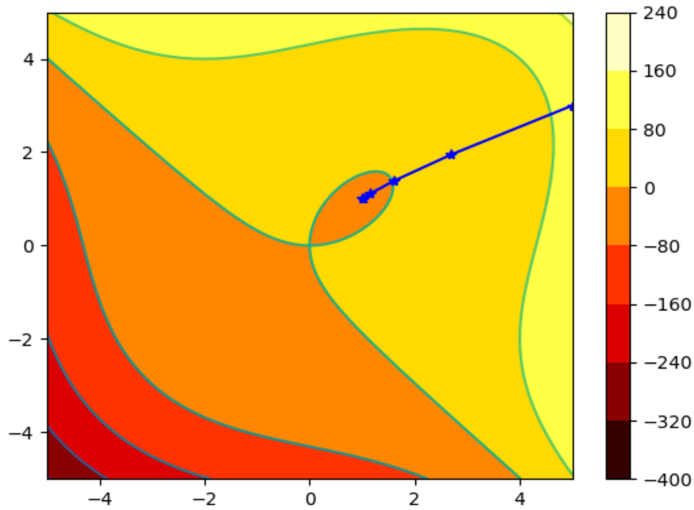

Figure 3: Newton method with initial $\mathbf{x} = (1, 2)^T$

可以发现经过短暂几步的迭代,Newton 法就停止了搜索, 探究其原因, 是因为在 $\mathbf{x} = (1, 1)^T$ 处函数 $\mathbf{f}$ 的梯度为 0, 此时 Newton 法不能再更新. 而 $\mathbf{x} = (1, 1)^T$ 点的 Hessian 矩阵为

$\begin{pmatrix}6 & -3\\ -3 & 6\end{pmatrix}$, 其一阶子式为 6, 二阶子式为 27, 故为正定矩阵, 此时 $\mathbf{x}=(1,1)^T$ 为极小值点. 这也可以看出 Newton 法极快的收敛速度. 如图 3:

但是在某些初值, 例如 $\mathbf{x}=(-2,-1)^T$ 处, 由于 Newton 法的学习率是固定的, 会出现在迭代的不同时期, 算法的步长出现过长或过小的情况, 如图 4:

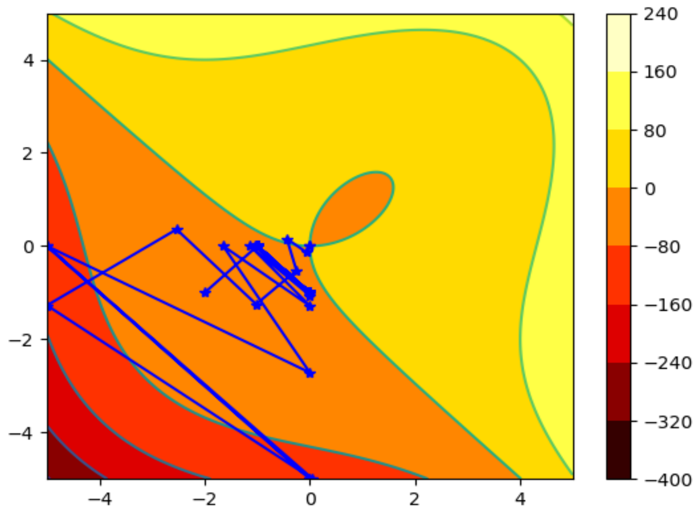

Figure 4: Newton method with initial $\mathbf{x}=(-2,-1)^T$

事实上, 引入自适应的步长能够解决这一问题, 例如 Adagrad 以及著名的 Adam 算法.

而对于最速下降法，由于每一步的搜索步长，或者说学习率，都相当于求解一个一维搜索问题，故是随着迭代的进行而不断变化的，因此可以找到正确的下降方向 (这里我加了边界限制，否则的话会跑到 $-\infty$...)，对于某些初值 (如 $\mathbf{x}=(-0.01,-0.03)^T$)，会趋向于" 全局最小值"$(-\infty,-\infty)$， 如图 5:

对于某些初值, 例如 $\mathbf{x}=(0.01,0.03)^T$, 该算法会收敛到局部极小值点, 如 $\mathbf{x}=(1,1)^T$, 如图 6:

但是无论是对于 Newton 法还是最速下降法，一旦初值选为 $(0,0)$，算法都不会进行；同时，若某一步到达 $(0,0)$ 点，算法也不会继续进行，因为 $(0,0)$ 是鞍点，要克服鞍点问题需要引入 momentum(动量). 此外, 由于在最速下降法中, 每次迭代都要执行一次一维搜索 (**尽管我采用了收敛较快的斐波那契法，同时通过先求解斐波那契表的方法提高了算法效率，但速度还是相对慢**), **算法的效率是较低的，梯度下降系列算法则可以很好的解决这一问题，其中考虑最为全面的亦是 Adam 算法，下将介绍并实现 Adam 算法 (但是显然，本例中还不涉及样本的概念).**

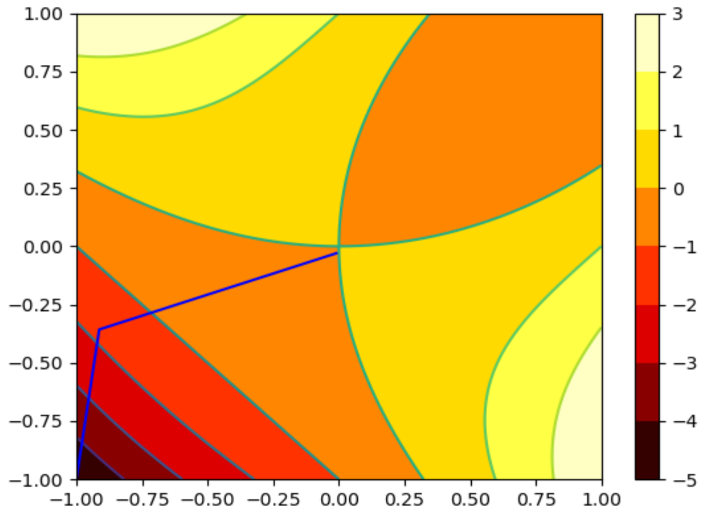

Figure 5: FastedDescent method with initial $\mathbf{x} = (-0.01, -0.03)^T$

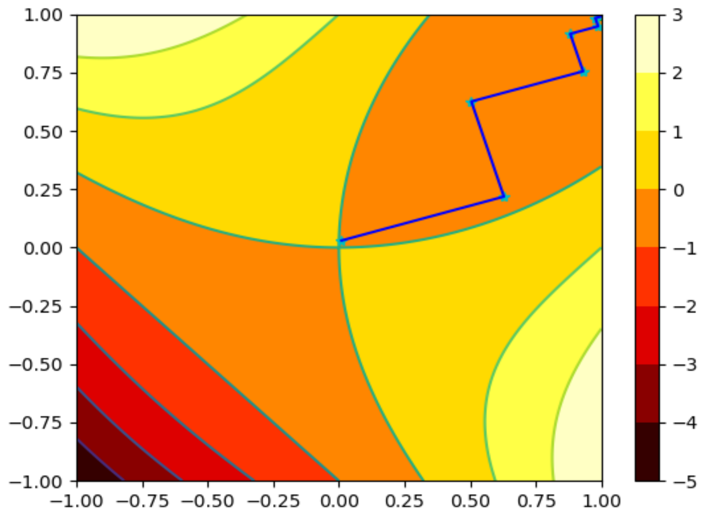

Figure 6: FastedDescent method with initial $\mathbf{x} = (0.01, 0.03)^T$

算法 1 Adam 算法

Input: 步长 ϵ.

Input: 矩估计的指数衰减速率,$\rho_1, \rho_2 \in [0, 1)$.

Input: 用于数值稳定的较小常数 δ.

Input: 初始参数 θ.

Input: 初始化一阶和二阶矩变量 $s = 0, r = 0$.

Input: 初始化时间步 $t = 0$.

while 没有达到停止准侧 **do**

  从训练集中采集包含 m 个样本 $x^1, \ldots, x^m$ 的小批量, 对应目标为 y^i.

  计算梯度:$\mathbf{g} \leftarrow \frac{1}{m}\nabla_\theta \sum_i L(f(x^i, \theta), y^i)$.

  $t \leftarrow t + 1$.

  更新有偏一阶估计:$\mathbf{s} \leftarrow \rho_1 \mathbf{s} + (1 - \rho_1)\mathbf{g}$.

  更新有偏二阶估计:$\mathbf{r} \leftarrow \rho_2 \mathbf{r} + (1 - \rho_2)\mathbf{g} \bigodot \mathbf{g}$.

  修正一阶矩的偏差:$\hat{\mathbf{s}} \leftarrow \frac{s}{1-\rho_1^t}$.

  修正二阶矩的偏差:$\hat{\mathbf{r}} \leftarrow \frac{r}{1-\rho_2^t}$.

  计算更新:$\Delta\theta = -\epsilon \frac{\hat{\mathbf{s}}}{\sqrt{\hat{\mathbf{r}}} + \delta}$.(逐元素应用操作)

  应用更新:$\theta \leftarrow \theta + \Delta\theta$.

end while

而在 ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION 原文中的伪代码如图 7:

最后展示一下 Adam 的结果, 它只用了 17 步就收敛到了接近 $(1, 1)$ 的位置:

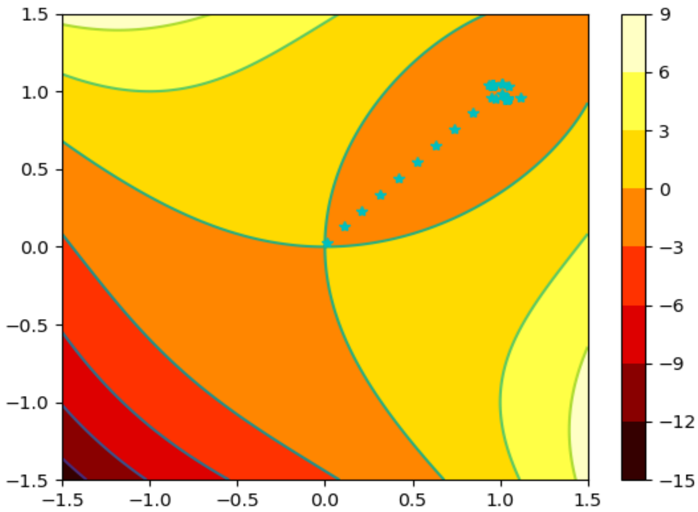

Figure 8: Adam method with initial $\mathbf{x} = (0.01, 0.03)^T$

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

Figure 7: Pseudocode of Adam in original paper

3 一点体会

初值选择真的很重要! 初值选择真的很重要! 初值选择真的很重要!

References

[1] 李航. 统计学习方法 [M]. 清华大学出版社, 2012.

[2] Kingma D , Ba J . Adam: A Method for Stochastic Optimization[J]. Computer Science, 2014.

[3] Goodfellow, Ian, et al. Deep Learning[M]. MIT Press, 2016.