Advances in Applied Mathematics 应用数学进展, 2022, 11(4), 2242-2258
Published Online April 2022 in Hans. http://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2022.114238

Hans汉斯

Barzilai-Borwein型算法的探讨

黄亚楠

长沙理工大学数学与统计学院，湖南 长沙

收稿日期：2022年3月26日；录用日期：2022年4月21日；发布日期：2022年4月28日

摘 要

本文介绍了八种负梯度算法，根据特点对其进行了比较，并对不同维数的严格凸二次函数进行了计算，绘制图表观察数据，发现BB型算法更具优势，维数越大优势越大。选取不同的初始步长可以改变算法的效果，特别是当矩阵条件数越大时，初始步长的选取越关键，本文考虑了四种选择初始步长的方法，分别将其放入算法中进行数值实验，结果表明，选取Hessian矩阵最小特征值的倒数效果最好。最后，介绍了负梯度算法在深度学习中的应用。

关键词

Barzilai-Borwein型算法，负梯度算法，初始步长，算法的应用

Discussion on Barzilai-Borwein Algorithm

Ya'nan Huang

School of Mathematics and Statistics, Changsha University of Science and Technology, Changsha Hunan

Received: Mar. 26th, 2022; accepted: Apr. 21st, 2022; published: Apr. 28th, 2022

Abstract

This paper introduces eight kinds of negative gradient algorithms, compares them according to their characteristics, calculates strictly convex quadratic functions of different dimensions, draws graphs and observes data, and finds that BB algorithm is more advantageous, the larger the dimension is, the greater the advantage is. Selecting different initial step size can change the effect of the algorithm, especially when the number of matrix conditions is larger, the selection of the initial step size is more critical. This paper considers four methods of selecting the initial step size, and puts them into the algorithm for numerical experiments. The results show that the reciprocal of the minimum eigenvalue of the Hessian matrix is the best. Finally, the application of negative gradient algorithm in deep learning is introduced.

文章引用: 黄亚楠. Barzilai-Borwein 型算法的探讨[J]. 应用数学进展, 2022, 11(4): 2242-2258.
DOI: 10.12677/aam.2022.114238

Keywords

Barzilai-Borwein Algorithm, Negative Gradient Algorithm, Initial Step Size, Application of the Algorithm

1. 引言

Barzilai-Borwein (BB)算法[1]是 1988 年提出的一种负梯度算法，它可以追溯到 1847 年 Cauchy 提出的最速下降法(SD) [2]，在当时产生了很大的影响。由于收敛速度慢，迭代次数过多，SD 算法并没有引起人们的重视，为了提高负梯度算法的收敛速度，减少了迭代次数，Barzilai 和 Borwein 提出了 BB 算法，并与 SD 进行了比较，给出了二次收敛和渐近收敛的证明，实验结果表明，BB 算法在严格凸二次函数的极小化方面具有更强的竞争力。

BB 算法提出后，负梯度算法越来越受到人们的关注。Raydan [3]等人利用位移幂法(计算矩阵特征值和特征向量)，得到了 BB 算法对 n 维严格凸二次函数具有全局收敛性。Dai [4]在 Raydan 等人的基础上进一步证明 BB 算法对 n 维严格凸二次函数收敛速度是 R-线性收敛的，虽然这一结果没有 SD 算法的 Q-线性速度，但 BB 算法的实际计算效果要远好于 SD 算法。在此基础上 Fletcher [5]等人通过忽略递归关系中的某些项，定义了该方法的简化版本，给出了 n 维 BB 算法的渐近收敛。

2003 年 Dai [6]等人在 BB 算法的基础上，提出了交替使用 BB 步长的交替步梯度法(AS)，并分析在 2 维的情况下 AS 算法是两步 Q-超线性收敛的，在 n 维情情况下 AS 算法是 R-线性的收敛性，对比 BB 算法发现当矩阵 A 维数越大，AS 算法的迭代次数比 BB 算法的迭代次数就越少，说明 AS 算法是一种很有潜力的 BB 型算法，为解决大规模问题提供了很好的选择。不仅如此在 2006 年预处理 BB 算法(ABB) [7]提出，实验结果表明 ABB 算法明显优于已有 BB 型算法。本文在此基础上分析了多种负梯度算法，并应用到了不同维数的严格凸二次函数中，分析比较了数值结果，发现初始步长选择合适可以减少迭代次数和时间，数值结果表明不同的初始步长产生的迭代次数不同，为求解大规模问题提供了新的选择，例如信号处理[8]、机器学习[9]和零售问题[10] [11] [12]。

2. Barzilai-Borwein 步长的选取

考虑严格凸二次函数的极小化问题：

$$\min_{x\in\mathbb{R}^n} f(x)=\frac{1}{2}x^{\mathrm{T}}Ax-b^{\mathrm{T}}x \tag{2.1}$$

中 BB 步长 α_k 是如何选取的，其中矩阵 $A\in\mathbb{R}^{n\times n}$ 是对称正定的，向量 $b\in\mathbb{R}^n$。利用 SD 算法和 BB 算法进行求解时，发现 SD 算法求解效果不好，尤其当 A 的条件数($cond(A)=\|A\|\|A^{-1}\|$)很大时，效率很低，而 BB 算法表现良好。

负梯度算法的迭代格式均为 $x_{k+1}=x_k-a_k\nabla f(x_k)$，其中 a_k 表示步长，$\nabla f(x_k)$ 表示梯度，经过计算梯度可以写成 $\nabla f(x_k)=Ax_k-b$。BB 型算法的基本思想源于拟牛顿算法，它希望步长 a_k 使矩阵 $D_k=\alpha_k I$ 满足拟牛顿方程：

$$D_k s_{k-1}=y_{k-1} \tag{2.2}$$

$$\text{记：}\ s_{k-1}=x_k-x_{k-1},\quad y_{k-1}=\nabla f\left(x_{k-1}\right)-\nabla f\left(x_{k-1}\right) \tag{2.3}$$

设 D_k 为函数 $f(x)$ 在 x_k 点处的 Hessian 矩阵 A_k 的逼近，利用上一步的迭代信息，使得矩阵 D_k 具有一定的拟牛顿性质，在二范数的意义下取极小：

$$\alpha_k=\arg\min_{\alpha_k>0}\left\|D_k^{-1}s_{k-1}-y_{k-1}\right\|_2^2 \tag{2.4}$$

对这个极小化问题求解，并把解记为 α_k^{BB1} 得到：

$$\alpha_k^{BB1}=\frac{s_{k-1}^{\mathrm{T}}s_{k-1}}{s_{k-1}y_{k-1}} \tag{2.5}$$

根据对称性使得矩阵 D_k 的逆 D_k^{-1} 也有一定的拟牛顿性质，同样的在二范数的意义下取极小：

$$\alpha_k=\arg\min_{\alpha_k>0}\left\|s_{k-1}-D_k y_{k-1}\right\|_2^2 \tag{2.6}$$

对这个极小化问题求解，并把解记为 α_k^{BB2} 得到：

$$\alpha_k^{BB2}=\frac{s_{k-1}^{\mathrm{T}}y_{k-1}}{y_{k-1}^{T}y_{k-1}} \tag{2.7}$$

通常把 α_k^{BB1} 称为长 BB 步长，把 α_k^{BB2} 称为短的 BB 步长利用 SD 算法中：

$$x_k-x_{k-1}=-\alpha_k\nabla f\left(x_k\right) \tag{2.8}$$

$$\nabla f\left(x_k\right)-\nabla f\left(x_{k-1}\right)=A\left(x_k-x_{k-1}\right)=A\left(-\alpha_k\nabla f\left(x_k\right)\right) \tag{2.9}$$

带入到 BB 算法得到：

$$s_{k-1}=x_k-x_{k-1}=-\alpha_k\nabla f\left(x_k\right) \tag{2.10}$$

$$y_{k-1}=\nabla f\left(x_k\right)-\nabla f\left(x_{k-1}\right)=A\left(x_k-x_{k-1}\right)=A\left(-\alpha_k\nabla f\left(x_k\right)\right) \tag{2.11}$$

那么 α_k^{BB1} 可以写成：

$$\alpha_k^{BB1}=\frac{\left(-\alpha_{k-1}\nabla f\left(x_{k-1}\right)\right)^{\mathrm{T}}\left(-\alpha_{k-1}\nabla f\left(x_{k-1}\right)\right)}{\left(-\alpha_{k-1}\nabla f\left(x_{k-1}\right)\right)^{\mathrm{T}}A\left(-\alpha_{k-1}\nabla f\left(x_{k-1}\right)\right)}=\frac{\nabla f\left(x_{k-1}\right)^{\mathrm{T}}\nabla f\left(x_{k-1}\right)}{\nabla f\left(x_{k-1}\right)^{\mathrm{T}}A\nabla f\left(x_{k-1}\right)} \tag{2.12}$$

同样的 α_k^{BB2} 可以写成：

$$\alpha_k^{BB2}=\frac{\left(-\alpha_{k-1}\nabla f\left(x_{k-1}\right)\right)^{\mathrm{T}}A\left(-\alpha_{k-1}\nabla f\left(x_{k-1}\right)\right)}{\left(-\alpha_{k-1}\nabla f\left(x_{k-1}\right)\right)^{\mathrm{T}}A^{\mathrm{T}}A\left(-\alpha_{k-1}\nabla f\left(x_{k-1}\right)\right)}=\frac{\nabla f\left(x_{k-1}\right)^{\mathrm{T}}A\nabla f\left(x_{k-1}\right)}{\nabla f\left(x_{k-1}\right)^{\mathrm{T}}A^2\nabla f\left(x_{k-1}\right)} \tag{2.13}$$

与 SD 算法和 MG (Minimum Gradient) [4]算法的步长相比：

$$\alpha_k^{SD}=\arg\min_{\alpha_k>0}f\left(x_k-\alpha\nabla f\left(x_k\right)\right)=\frac{\nabla f\left(x_k\right)^{\mathrm{T}}\nabla f\left(x_k\right)}{\nabla f\left(x_k\right)^{\mathrm{T}}A\nabla f\left(x_k\right)} \tag{2.14}$$

和

$$\alpha_k^{MG}=\arg\min_{\alpha_k>0}f\left(x_k-\alpha\nabla f\left(x_k\right)\right)=\frac{\nabla f\left(x_k\right)^{\mathrm{T}}A\nabla f\left(x_k\right)}{\nabla f\left(x_k\right)^{\mathrm{T}}A^2\nabla f\left(x_k\right)} \tag{2.15}$$

BB 步长是取 SD 算法或 MG 算法的上一步步长，在实际计算中，BB 算法的数值结果明显优于 SD 算法和 MG 算法。BB 型算法的迭代过程中，函数值并不严格减小。数值实验表明，BB 型算法实际上是一种非单调下降方法，在求解严格凸二次函数时具有优势。

3. 算法介绍

负梯度算法设计较为简单，计算量小，储存变量少，对初始点没有特别要求，许多算法的初始方向都是最速下降方向(即负梯度方向)。由于 BB 步长没有严格的规定，BB 型算法也逐渐增多，接下来将普通的负梯度法和 BB 型算法两两分组进行介绍。

3.1. 最速下降法和 Barzilai-Borwein-1 算法

最速下降法是一种最古老的优化算法，它的收敛速度与目标函数的性质有极强相关性，例如矩阵的维数，矩阵的条件数，但其算法简单以目标函数的负梯度方向为搜索方向，具体算法如下：

算法 1 (SD 算法)：令 $\{x_k\}$ 为 SD 算法求解问题(2.1)过程中的迭代序列

步骤 1 初始化参数：$x_0 \in \Re^n$，误差 ε，$k=1$，最大迭代次数 $iMax$，初始步长 a_1；

步骤 2 收敛性检验：若 $\left\|\nabla f\left(x_k\right)\right\| \le \varepsilon$，输出 x_k 作为近似的数值解；

步骤 3 确定下降搜索方向：计算 $d_k = -\nabla f\left(x_k\right)$；

步骤 4 选取步长：$\alpha_k = \dfrac{\nabla f\left(x_k\right)^{\mathrm{T}} \nabla f\left(x_k\right)}{\nabla f\left(x_k\right)^{\mathrm{T}} A \nabla f\left(x_k\right)}$；

步骤 5 定义下一步迭代：取 $x_{k+1} = x_k - a_k \nabla f\left(x_k\right)$，$k = k+1$ 转步骤 2。

SD 算法每次迭代的计算量和储存量少，对初始点的要求不高，可以比较快地从初始点到达极小点附近，但在接近极小点时会出现锯齿现象，所以每次迭代移动的步长很小，具有明显的周期性，侧面说明该下降方向是规则出现的，使收敛速度很慢，BB-1 算法继承了 SD 算法的优点同时避免了锯齿现象的产生，由于 BB-1 步长的特点，函数值并不严格减小，但收敛速度快和迭代次数少，在求解严格凸二次函数时具有优势。

算法 2 (BB-1 算法)：令 $\{x_k\}$ 为 BB-1 算法求解问题(2.1)过程中的迭代序列，

步骤 1 初始化参数：$x_0 \in \Re^n$，误差 ε，$k=1$，最大迭代次数 $iMax$，初始步长 a_1；

步骤 2 收敛性检验：若 $\left\|\nabla f\left(x_k\right)\right\| \le \varepsilon$，输出 x_k 作为近似的数值解；

步骤 3 确定下降搜索方向：计算 $d_k = -\nabla f\left(x_k\right)$；

步骤 4 选取步长：$\alpha_k = \dfrac{s_{k-1}^{\mathrm{T}} s_{k-1}}{s_{k-1} y_{k-1}}$；

步骤 5 定义下一步迭代：取 $x_{k+1} = x_k - \alpha_k \nabla f\left(x_k\right)$，$k = k+1$ 转步骤 2。

3.2. 最小梯度法和 Barzilai-Borwein-2 算法

最小梯度法(MG)是在 SD 算法的基础上使函数值 $f(x)$ 沿着射线 $\left\{x_k - \alpha_k \nabla f\left(x_k\right) : \alpha > 0\right\}$ 最小化，一定程度上加速了收敛速度，减少了迭代次数，但同样会在接近极小点时出现锯齿现象，而 BB-2 算法继承了 MG 算法优点的同时，避免了锯齿现象加速了收敛速度，减少了迭代次数和迭代时间。

算法 3 (MG 算法)：令 $\{x_k\}$ 为 MG 算法求解问题(2.1)过程中的迭代序列

步骤 1 初始化参数：$x_0 \in \Re^n$，误差 ε，$k=1$，最大迭代次数 $iMax$，初始步长 a_1；

步骤 2 收敛性检验：若 $\left\|\nabla f\left(x_k\right)\right\| \le \varepsilon$，输出 x_k 作为近似的数值解；

步骤 3 确定下降搜索方向：计算 $d_k = -\nabla f\left(x_k\right)$；

步骤 4 选取步长：$\alpha_k = \dfrac{\nabla f(x_k)^{\mathrm{T}} A \nabla f(x_k)}{\nabla f(x_k)^{\mathrm{T}} A^2 \nabla f(x_k)}$；

步骤 5 定义下一步迭代：取 $x_{k+1} = x_k - a_k \nabla f(x_k)$，$k = k+1$ 转步骤 2。

算法 4 (BB-2 算法)：令 $\{x_k\}$ 为 BB-2 算法求解问题(2.1)过程中的迭代序列，

步骤 1 初始化参数：$x_0 \in \Re^n$，误差 ε，$k=1$，最大迭代次数 $iMax$，初始步长 a_1；

步骤 2 收敛性检验：若 $\|\nabla f(x_k)\| \le \varepsilon$，输出 x_k 作为近似的数值解；

步骤 3 确定下降搜索方向：计算 $d_k = -\nabla f(x_k)$；

步骤 4 选取步长：$\alpha_k = \dfrac{s_{k-1}^{\mathrm{T}} y_{k-1}}{y_{k-1}^{\mathrm{T}} y_{k-1}}$；

步骤 5 定义下一步迭代：取 $x_{k+1} = x_k - a_k \nabla f(x_k)$，$k = k+1$ 转步骤 2。

3.3. 交替最小化梯度法和交替梯度法

交替最小化梯度法(AM)是交替使用步长 α_k^{SD} 和 α_k^{MG}，受矩阵条件数的影响较小，交替梯度法(AS)是在经典 BB 算法的基础上，交替使用步长 α_k^{SD} 和 α_k^{BB1}，这两种算法在接近极小点时不会出现锯齿现象，在求解大规模问题时具有优势。

算法 5 (AM 算法)：令 $\{x_k\}$ 为 AM 算法求解问题(2.1)过程中的迭代序列，

步骤 1 初始化参数：$x_0 \in \Re^n$，误差 ε，$k=1$，最大迭代次数 $iMax$，初始步长 a_1；

步骤 2 收敛性检验：若 $\|\nabla f(x_k)\| \le \varepsilon$，输出 x_k 作为近似的数值解；

步骤 3 确定下降搜索方向：计算 $d_k = -\nabla f(x_k)$；

步骤 4 选取步长：当 k 为奇数时 $\alpha_k = \dfrac{\nabla f(x_k)^{\mathrm{T}} \nabla f(x_k)}{\nabla f(x_k)^{\mathrm{T}} A \nabla f(x_k)}$，当 k 为偶数时 $\alpha_k = \dfrac{\nabla f(x_k)^{\mathrm{T}} A \nabla f(x_k)}{\nabla f(x_k)^{\mathrm{T}} A^2 \nabla f(x_k)}$；

步骤 5 定义下一步迭代：取 $x_{k+1} = x_k - a_k \nabla f(x_k)$，$k = k+1$ 转步骤 2。

算法 6 (AS 算法)：令 $\{x_k\}$ 为 AS 算法求解问题(2.1)过程中的迭代序列，

步骤 1 初始化参数：$x_0 \in \Re^n$，误差 ε，$k=1$，最大迭代次数 $iMax$，初始步长 a_1；

步骤 2 收敛性检验：若 $\|\nabla f(x_k)\| \le \varepsilon$，输出 x_k 作为近似的数值解；

步骤 3 确定下降搜索方向：计算 $d_k = -\nabla f(x_k)$；

步骤 4 选取步长：当 k 为奇数时 $\alpha_k = \dfrac{\nabla f(x_k)^{\mathrm{T}} \nabla f(x_k)}{\nabla f(x_k)^{\mathrm{T}} A \nabla f(x_k)}$，当 k 为偶数时 $\alpha_k = \dfrac{s_{k-1}^{\mathrm{T}} s_{k-1}}{s_{k-1} y_{k-1}}$；

步骤 5 定义下一步迭代：取 $x_{k+1} = x_k - a_k \nabla f(x_k)$，$k = k+1$ 转步骤 2。

3.4. 预处理 SD 算法和预处理 BB 算法

预处理 SD 算法结合了 SD 算法和 MG 算法的优点，对两者步长进行判选择，从而使迭代次数减少加快函数下降，选择不同的 $\kappa, \delta \in (0,1)$，迭代次数不同。预处理 BB 算法是在预处理最速下降法(ASD)的基础上提出的，它结合了 BB-1 算法和 BB-2 算法的优点，对两者步长进行判选择，从而使迭代次数减少加快函数下降，选择不同的 $\mu \in (0,1)$，迭代次数不同，为解决大规模问题提供选择。

算法 7 (ASD 算法)：令 $\{x_k\}$ 为 ASD 算法求解问题(2.1)过程中的迭代序列，

步骤 1 初始化参数：$x_0 \in \Re^n$，误差 ε，$k=1$，最大迭代次数 $iMax$，初始步长 a_1，$\kappa, \delta \in (0,1)$；

步骤 2 收敛性检验：若 $\|\nabla f(x_k)\| \le \varepsilon$，输出 x_k 作为近似的数值解；

步骤 3 确定下降搜索方向：计算 $d_k = -\nabla f(x_k)$；

步骤 4 选取步长：若 $\alpha_k^{MG}/\alpha_k^{SD}>\kappa$， $\alpha_k=\alpha_k^{MG}$，否则 $\alpha_k=\alpha_k^{SD}-\delta\alpha_k^{MG}$；

步骤 5 定义下一步迭代：取 $x_{k+1}=x_k-a_k\nabla f\left(x_k\right)$， $k=k+1$ 转步骤 2。

算法 8 (ABB 算法)：令 $\{x_k\}$ 为 ABB 算法求解问题(2.1)过程中的迭代序列，

步骤 1 初始化参数：$x_0\in\Re^n$，误差 ε， $k=1$，最大迭代次数 $iMax$，初始步长 a_1， $\mu\in(0,1)$；

步骤 2 收敛性检验：若 $\left\|\nabla f\left(x_k\right)\right\|\le\varepsilon$，输出 x_k 作为近似的数值解；

步骤 3 确定下降搜索方向：计算 $d_k=-\nabla f\left(x_k\right)$；

步骤 4 选取步长：$\alpha_k^{BB2}/\alpha_k^{BB1}<\mu$， $\alpha_k=\alpha_k^{BB2}$，否则 $\alpha_k=\alpha_k^{BB1}$；

步骤 5 定义下一步迭代：取 $x_{k+1}=x_k-a_k\nabla f\left(x_k\right)$， $k=k+1$ 转步骤 2。

此外还有很多关于 BB 步长的研究例如使 $\alpha_k=\alpha_k^{NBB}=\sqrt{\alpha_k^{BB1}\alpha_k^{BB2}}$ 的新 BB [13]算法、在循环 BB 算法(CBB)基础上使 $\alpha_k=\alpha_k^{CABB}\begin{cases}\alpha_k^{BB2} & \alpha_k^{BB2}/\alpha_k^{BB1}<\kappa\\ \alpha_k^{CBB}\end{cases}$ 的自适应循环 BB [8] (CABB)算法、令 $\alpha_k^{stab}=\dfrac{\left\|x_{k+1}-x_k\right\|}{\left\|-\nabla f\left(x_k\right)\right\|}$ 使 $\alpha_k=\min\left\{\alpha_k^{BB},\alpha_k^{stab}\right\}$ 的 BBstab [9]算法，这些都是在经典 BB 算法的基础上优化 BB 步长进而产生的新算法，这些都是在改变迭代步长，但数值研究表明，初始 α_1 的选取可以影响算法的迭代次数和运行时间，选取初始步长的方法如下：

1) 采用任一种一维寻优法，此时 $\alpha_k=\arg\min\limits_{\alpha_k>0}f\left(x_k-\alpha\nabla f\left(x_k\right)\right)$ 中 $f\left(x_k-\alpha\nabla f\left(x_k\right)\right)$ 是 α 唯一元函数，利用任一种一维寻优都可以求得 α_1。

2) 微分法，由 $\alpha f\left(x_k-\alpha\nabla f\left(x_k\right)\right)=\varphi(\alpha)$ 对于一些简单的情况可以直接令 $\varphi'(\alpha)=0$ 求出近似的最优步长 α_1 的值。

3) 直接赋值，可以直接令 $\alpha_1=1$。

4) 选取 Hessian 矩阵最小或最大特征值的倒数 $\dfrac{1}{\lambda\min}$ 或 $\dfrac{1}{\lambda\max}$。

利用上述介绍的算法解决不同维数的严格凸二次函数，同时利用上述方法选取初始步长 α_1 进行数值实验，观察算法产生的迭代次数、迭代时间、函数值的变化。

4. 数值实验与比较

本节通过上述八种优化算法求解(2.1)进行数值实验，为了避免偶然性造成的误差，随机生成(0, 100)内的 2、4、10、20、30 个正整数作为 $\{\lambda_1,\lambda_2,\cdots,\lambda_n\}$ 令(2.1)中 $A=diag\left(\lambda_1,\lambda_2,\cdots,\lambda_n\right)$ 和 $b=e_n$，选取不同的初始步长 α_1 观察迭代次数、迭代时间。设置相关参数误差 $\varepsilon=10^{-8}$、$iMax=10000$、$\kappa=0.5$、$\delta=0.5$、$\mu=0.5$、 $x_1=0$。 $A_1=diag(30,2)$、 $b_1=e_2$； $A_2=diag(82,89,34,71)$、 $b_2=e_4$；
$A_3=diag(89,56,9,69,95,5,97,13,61,86)$、 $b_3=e_{10}$；
$A_4=diag(80,61,25,37,53,21,92,37,5,52,60,64,34,45,75,38,61,99,61,88)$、 $b_4=e_{20}$；
$A_5=diag(13,9,20,11,19,50,99,98,76,71,17,21,28,23,20,53,48,48,18,76,63,27,67,73,1,21,59,67,60,60)$ 、
$b_5=e_{30}$；求得 A_1、 A_2、 A_3、 A_4、 A_5 的条件数分别为 15、2.62、19.4、19.8、99。

本文所有实验在 Windows 操作系统(主频 2.59 GHz 的 64 位 CPU，运行内存 8 GB)平台上 MatlabR2018b 软件和 Jupyter Notebook (py34)的环境中进行。方法一取初始步长 $\alpha_1=\dfrac{\nabla f\left(x_1\right)^{\mathrm{T}}\nabla f\left(x_1\right)}{\nabla f\left(x_1\right)^{\mathrm{T}}A\nabla f\left(x_1\right)}$，方法二取初始步长 $\alpha_1=1$，方法三取初始步长 $\alpha_1=\dfrac{1}{\lambda_{\min}}$，方法四取初始步长 $\alpha_1=\dfrac{1}{\lambda_{\max}}$。利用 python 语言编写程序整理出八种算法选取不同初始步长的迭代次数，如表 1 所示：

Table 1. The number of iterations of 8 algorithms
表 1. 8 种算法的迭代次数

维数 n \ 算法	SD	BB-1	MG	BB-2	AM	AS	ASD	ABB
2	142	10	47	122	10	84	47	10
2	13	9	9	9	12	7	9	9
2	3	4	3	4	3	3	3	4
2	3	4	2	4	3	3	2	4
4	24	14	25	14	15	13	24	14
4	29	15	30	18	21	15	29	15
4	11	10	11	10	11	9	11	10
4	19	15	20	14	15	13	19	15
10	178	43	118	42	75	47	76	43
10	198	48	186	49	95	54	78	48
10	102	38	97	34	45	36	58	38
10	156	44	176	44	67	48	67	44
20	187	45	116	44	61	57	65	45
20	203	55	200	53	85	61	76	55
20	49	27	49	26	33	27	49	27
20	173	49	171	53	65	49	79	49
30	931	105	857	91	320	128	140	105
30	118	45	116	44	132	44	67	45
30	118	45	116	44	132	44	67	45
30	880	109	900	104	252	118	137	109

整理出程序运行所需要的时间，为了更方便地观察将时间扩大 1000 倍，如表 2 所示：

Table 2. Iteration time of 8 algorithms
表 2. 8 种算法的迭代时间

维数 n \ 算法	SD	BB-1	MG	BB-2	AM	AS	ASD	ABB
2	325	21	85	84	16	60	76	7
2	205	9	9	6	7	3	7	3
2	51	4	7	6	3	2	4	6
2	7	6	5	5	3	4	4	4
4	13	10	14	10	18	14	17	10
4	13	7	13	10	13	9	21	9
4	7	8	11	12	13	12	9	7
4	10	9	11	8	7	7	22	8
10	124	30	87	24	50	26	67	37
10	214	28	138	36	56	24	67	28
10	82	31	74	27	31	18	67	25
10	99	22	125	34	41	24	64	31
20	143	65	163	29	91	80	123	45
20	150	37	275	52	67	36	109	51
20	40	25	41	20	41	15	125	28
20	205	33	228	38	63	57	79	44
30	1347	89	1492	69	354	151	311	139
30	115	68	175	37	208	61	189	44
30	115	68	175	37	208	61	189	44
30	1161	82	1237	117	359	85	310	138

将这八种算法的迭代次数绘图，如图 1~8 所示：

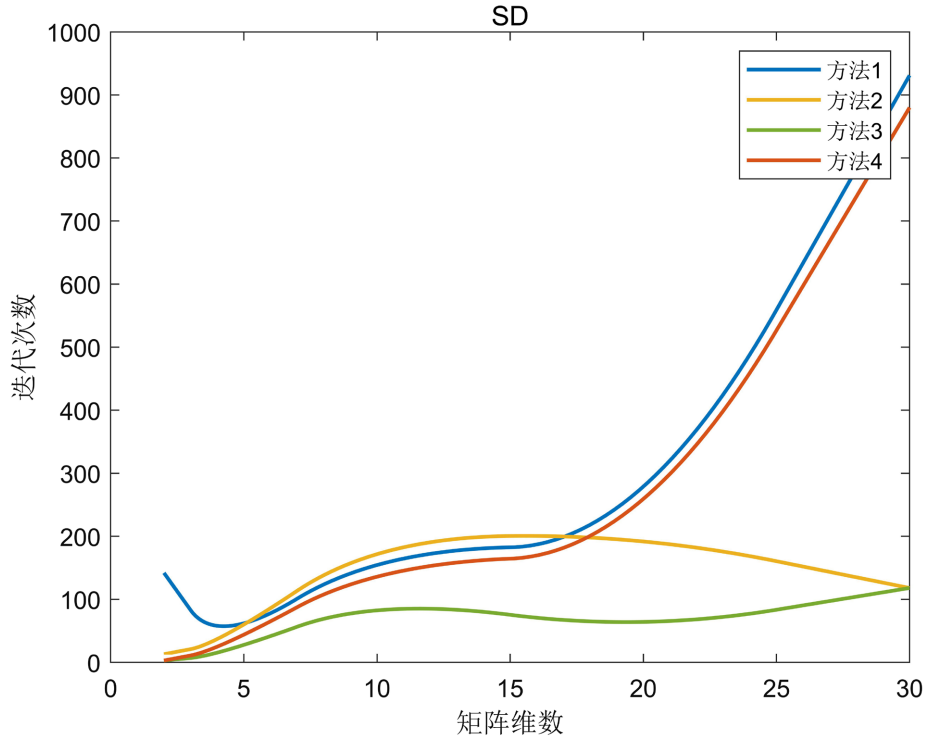

Figure 1. Gradient descent
图 1. 最速下降法

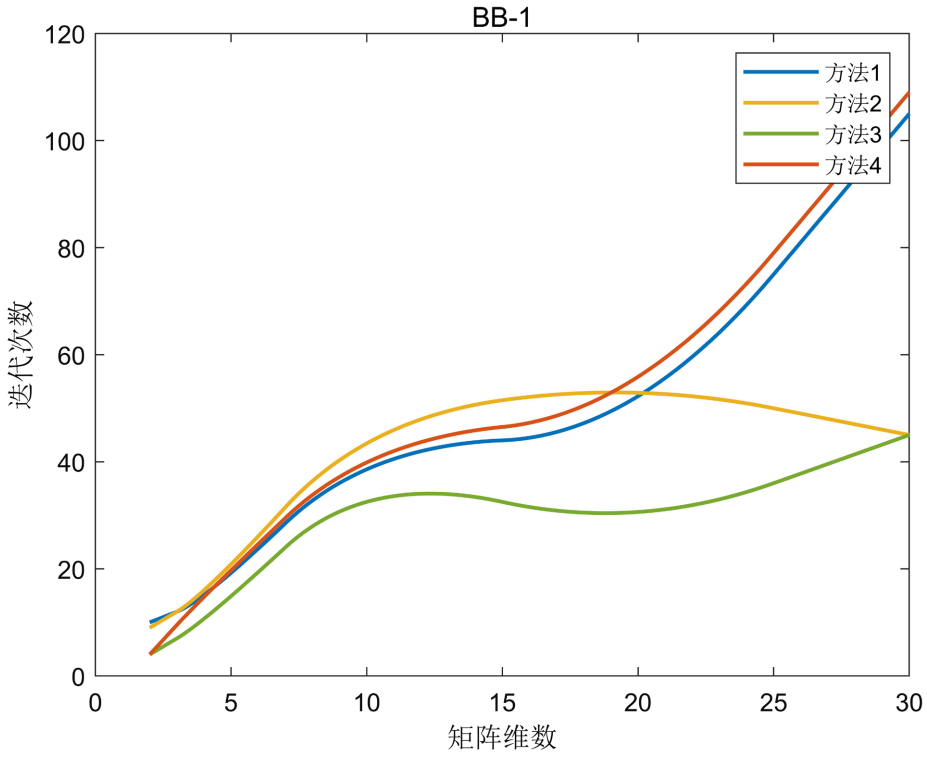

Figure 2. Barzilai-Borwein-1 algorithm
图 2. Barzilai-Borwein-1 算法

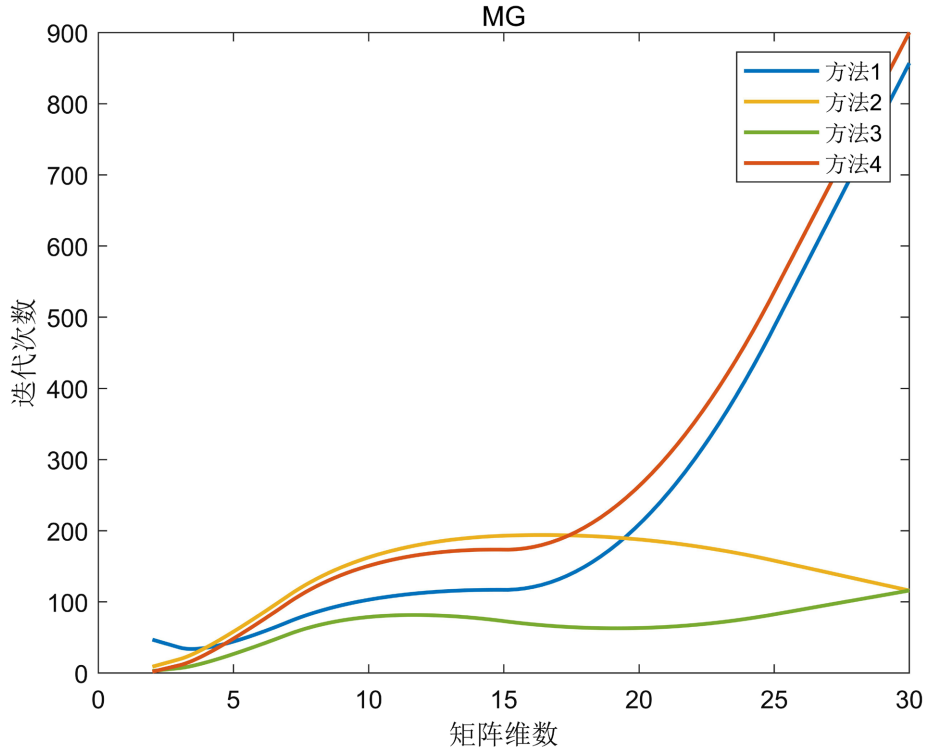

Figure 3. Minimum gradient method
图 3. 最小梯度法

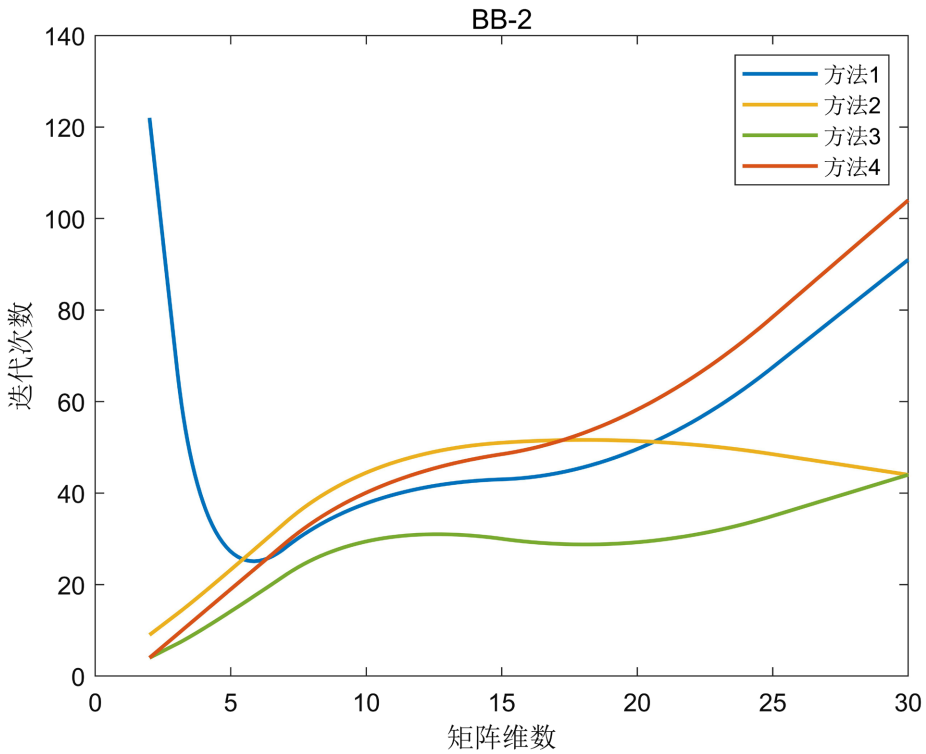

Figure 4. Barzilai-Borwein-2 algorithm
图 4. Barzilai-Borwein-2 算法

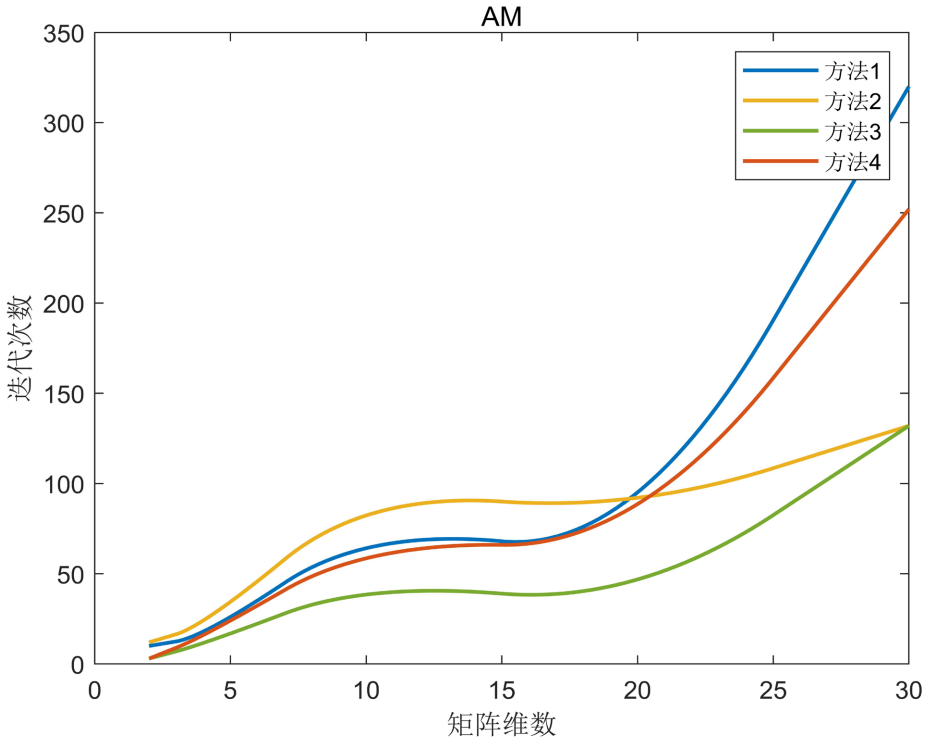

Figure 5. Alternating minimization gradient method
图 5. 交替最小化梯度法

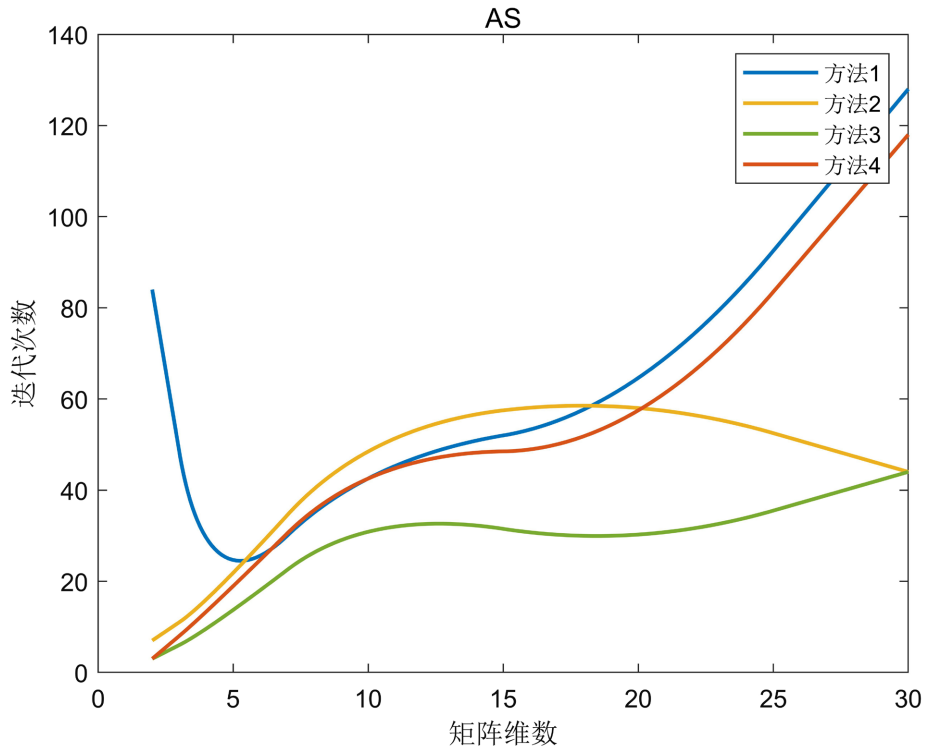

Figure 6. Alternating gradient method
图 6. 交替梯度法

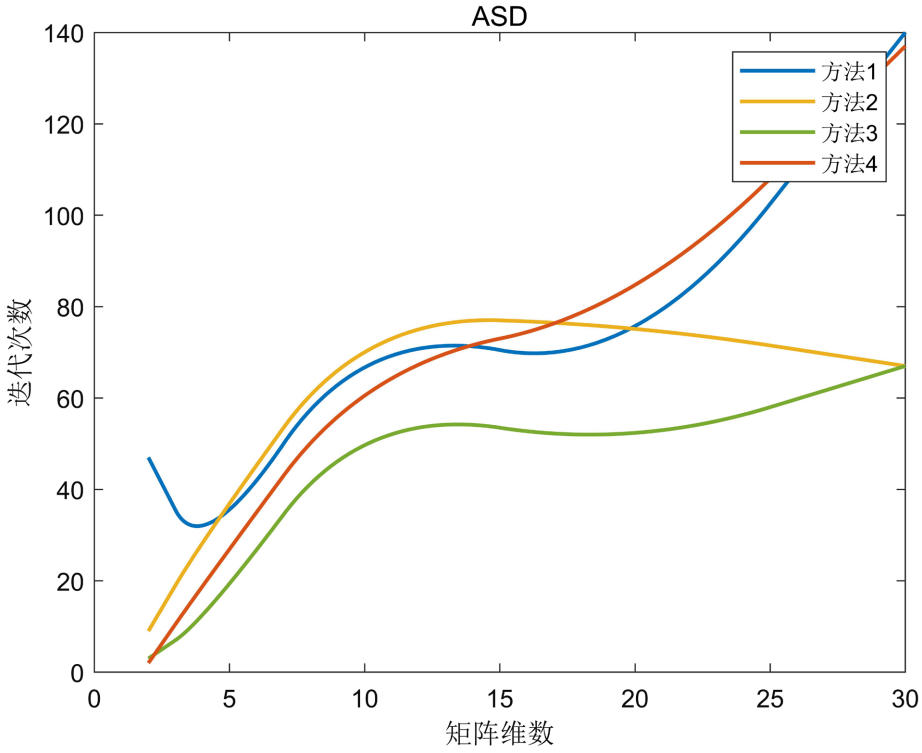

Figure 7. The fastest descent method of pretreatment
图 7. 预处理最速下降法

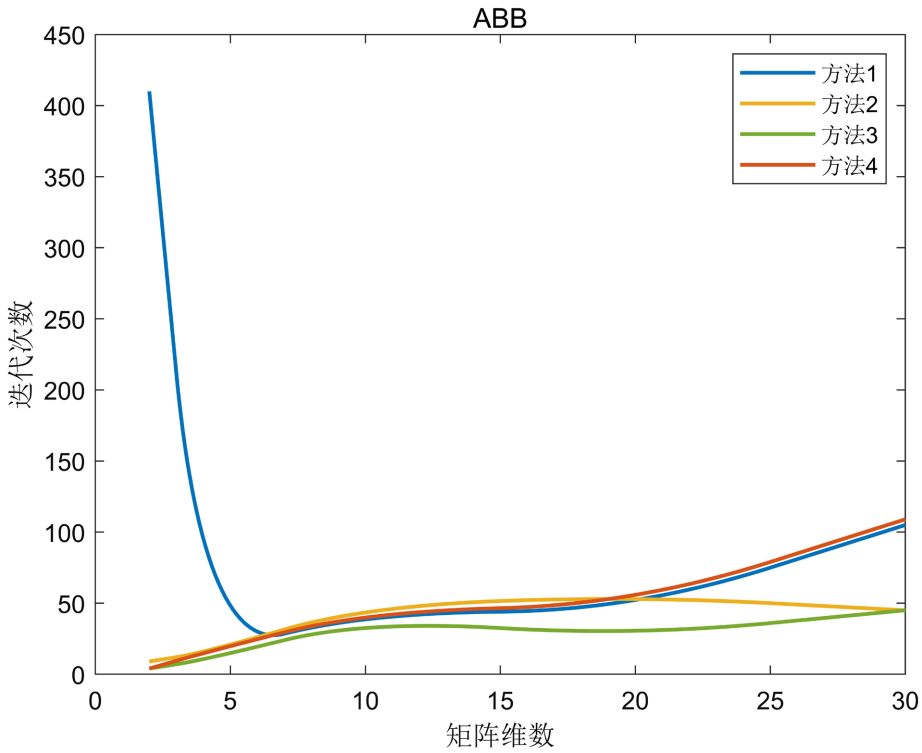

Figure 8. Preprocessing Barzilai-Borwein algorithm
图 8. 预处理 Barzilai-Borwein 算法

从图中可以看出，条件数对一般的负梯度算法影响较大，而 BB 算法更稳定，在解决大规模问题时更有竞争力。BB-1 与 BB-2 算法相比，后者在矩阵维数较小时没有明显的优势，但随着矩阵维数的增加差距逐渐减小，甚至有稍微的优势。ASD 算法对步长进行了选择，迭代次数大大减少，且当矩阵条件数越大其优势越发明显。ABB 算法对 BB 步长进行了选择，实验结果表明，当矩阵维数较小时，ABB 算法更接近 BB-2 算法；当矩阵维数较大时，ABB 算法更接近 BB-1 算法。BB 型算法比一般的负梯度算法更复杂，计算和存储的变量更多，但在运行时间上仍占优势。实验数据表明初始步长选择方法一和方法四的效果相似，方法三的效果最好，特别是当矩阵的维数和条件数越大时越有优势，这也反映出选择合适的初始步长可以改变算法的效果。

5. 负梯度算法的应用

负梯度算法广泛应用于图像重建[14]、超分辨率图像技术[15]、文本识别等大规模问题。其中图像重建的方法主要分为两类：第一种是直接利用数学反计算求横截剖面，称为解析图像重建；第二种是一系列的区域迭代，称为迭代图像重建。迭代法能够充分利用系数的稀疏性，具有占用内存少、程序设计简单、无误差积累等优点，图像重建的求解方案：

步骤 1：将需要重建的图像放入直角网格，利用发射源和探测器进行扫描。

步骤 2：将每个像素按照扫描次序排列，第 j 个像素中 x_j 表示射线吸收系数，x_j 为常数，第 j 个像素与第 i 条射线相交的长度用 α_{ij} 表示，α_{ij} 代表第 j 个像素沿第 i 条射线的贡献值，用 y_i 性表示沿第 i 方条射线方向的总吸收值，写成线性方程 $y_i = \sum_{j=1}^{N} \alpha_{ij} x_j, i = 1,2,\cdots,N$ 。

步骤 3：将线性方程组写成矩阵形式 $AX = Y$，其中 X, Y 都是向量 A 是投影矩阵。

步骤 4：将 $AX = Y$ 转化为极小化问题 $\min f(x) = \|AX - Y\|_2^2$。

步骤 5：利用负梯度算法求解极小化目标函数 $f(x)$。对 CT (Computed tomography)图 9 进行图像重建，结果如图 10 所示：

Figure 9. The original image
图 9. 原图像

Figure 10. The processed image
图 10. 处理后的图像

可以看出，经过处理的图像显示了原图像中不容易观察到的地方。虽然图像有一定的分辨率，但仍然不够清晰。超分辨率图像技术是通过优化算法将低分辨率图像转化为高分辨率图像。高分辨率图像具有较高的像素密度，使图像质量更细腻、更自然、纹理更清晰，这是图像增强领域中一个非常重要的问题。传统的超分辨率技术通常采用负梯度算法求解，求解方案如下：

步骤 1：假设模型，一般假设超分辨率模型为 $Y_k = D_k H_k F_k X + V_k, k = 1,2,\cdots,N$，其中向量 X 表示待求的高分辨率图像；向量 Y_k 表示所观察的低分辨率图像；D_k, F_k, H_k 分别表示下采样矩阵、模糊矩阵、运动矩阵；V_k 表示附加的高斯噪声。

步骤 2：在超分辨率模型中将这些估计量的定义重写为最小化问题

$\hat{X} = \arg\min_X \left[\sum_{k=1}^{N} \rho\left(Y_k, D_k H_k F_k X\right) \right], k = 1,2,\cdots,N$，其中 ρ 为模型和测量值之间的“距离”，令 $D_k H_k F_k = W_k, k = 1,2,\cdots,N$，当 ρ 是残差的二范数时可以得到最小二乘方程

$\hat{X} = \arg\min_X \left[\sum_{k=1}^{N} \left\| W_k X - Y_k \right\|_2^2 \right], k = 1,2,\cdots,N$。

步骤 3：为了避免因低分辨率图像提供数据不足而造成的伪影，引入正则化项 $\lambda J(X)$，那么问题将改写为 $\hat{X} = \arg\min_X \left[\sum_{k=1}^{N} \left\| W_k X - Y_k \right\|_2^2 + \lambda J(X) \right], k = 1,2,\cdots,N$，令 $\sum_{k=1}^{N} \left\| W_k X - Y_k \right\|_2^2 = E_\rho(X)$ 可以写成 $\hat{X} = \arg\min_X f(X) = \arg\min_X \left\{ E_\rho(X) + \lambda J(X) \right\}$。

步骤 4：利用负梯度算法求解极小化目标函数 $f(X)$。对低分辨率图 11 使用超分辨率技术，结果如图 12 所示：

Figure 11. The original image
图 11. 原图像

Figure 12. The processed image
图 12. 处理后的图像

原图像的分辨率为：250 × 250 而处理后的图像为：434 × 363，不仅如此，图像的大小也从 40.6 KB 减小到了 11.7 KB。超分辨率技术不仅提高了图像的分辨率，还减少了图像占用的内存。虽然图像的分辨率有所提高，但如何捕捉图像中的文字信息呢？字符识别技术适用于纸张打印、图片等字符的识别，识别结果以文本的形式存储在计算机中，如车牌识别就可以将优化算法和 BP 神经网络结合，可以选择

负梯度算法来寻找最优的连接权值，这样可以提高字符识别算法的准确率，达到预期的效果。

最优化问题在深度学习中占有非常重要的地位，很多深度学习模型最终都会归结为求解最优化问题，求解大规模最优化问题通常采用数值计算方法，而负梯度算法就是最经典的数值计算方法，具有存储容量小，结构简单，易于实现的优点，是最简便也是最常用的优化算法之一，在训练类算法中，负梯度算法通常用于求解极小化问题。在现实生活中，需要根据不同的数据选择合适的负梯度算法，以降低计算成本，提高算法的整体效率。

6. 结论与期望

随着计算机技术和算法语言的快速发展，各种优化算法不断被提出，为解决大规模问题提供了新的选择。本文对八种负梯度算法进行了讨论和比较。同时，将这些算法用于解决不同维数的严格凸二次函数。通过对比实验数据，发现 BB 型算法在迭代次数和迭代时间上都比一般的负梯度算法更有优势，并且矩阵的条件数越大优势越大。在数值实验中，发现选择不同的初始步长会产生不同的效果，通过比较发现 Hessian 矩阵最小特征值的倒数是最佳选择。最后，介绍了负梯度算法在图像重建、超分辨率图像技术、文本识别中的应用。

由于 BB 步长没有严格的规定，新的 BB 型算法不断被提出，理论知识不断被完善，它们在解决特殊问题时具有或多或少的优势。许多书籍在介绍优化算法时对 BB 型算法的介绍较为笼统，网上关于 BB 型算法的语言代码也不多，大家对 BB 型算法只有一个大概的印象，希望通过本文大家对 BB 型算法的发展、理论知识、特点有一个详细的认识。本文推广了 BB 型算法，为 BB 型算法的进一步研究提供了参考，后续将会对 BB 步长和共轭梯度进行研究，结合两者的优点设计出一种新的 BB 型算法。

参考文献

[1] Barzilai, J. and Borwein, J.M. (1988) Two-Point Step Size Gradient Methods. *IMA Journal of Numerical Analysis*, **8**, 141-148. https://doi.org/10.1093/imanum/8.1.141

[2] Cauchy, A. (1847) Méthode générale pour la résolution des systemes déquations simultanées. *Comptes rendus de l'Académie des Sciences Paris*, **25**, 536-538.

[3] Raydan, M. (1993) On the Barzilai and Borwein Choice of Steplength for the Gradient Method. *IMA Journal of Numerical Analysis*, **13**, 321-326. https://doi.org/10.1093/imanum/13.3.321

[4] Dai, Y.H. and Liao, L.Z. (2002) R-Linear Convergence of the Barzilai and Borwein Gradient Method. *IMA Journal of Numerical Analysis*, **22**, 1-10. https://doi.org/10.1093/imanum/22.1.1

[5] Roger, F. (2005) On the Barzilakborwein. *Optimization and Control with Applications*, **96**, 235-256.

[6] Dai, Y.H. (2003) Alternate Step Gradient Method. *Optimization*, **52**, 395-415. https://doi.org/10.1080/02331930310001611547

[7] Zhou, B., Gao, L. and Dai, Y.H. (2006) Gradient Methods with Adaptive Step-Sizes. *Computational Optimization and Applications*, **35**, 69-86. https://doi.org/10.1007/s10589-006-6446-0

[8] Wan, Z., Guo, J., Liu, J. and Liu, W. (2018) A Modified Spectral Conjugate Gradient Projection Method for Signal Recovery. *Signal, Image and Video Processing*, **12**, 1455-1462. https://doi.org/10.1007/s11760-018-1300-2

[9] Sopyła, K. and Drozda, P. (2015) Stochastic Gradient Descent with Barzilai-Borwein Update Step for SVM. *Information Sciences*, **316**, 218-233. https://doi.org/10.1016/j.ins.2015.03.073

[10] Chen, X., Liu, Y. and Wan, Z. (2016) Optimal Decision Making for Online and Offline Retailers under BOPS Mode. *The ANZIAM Journal*, **58**, 187-208. https://doi.org/10.1017/S1446181116000201

[11] Li, Y., Wan, Z. and Liu, J. (2017) Bi-Level Programming Approach to Optimal Strategy for Vendor-Managed Inventory Problems under Random Demand. *The ANZIAM Journal*, **59**, 247-270. https://doi.org/10.1017/S1446181117000384

[12] Zhang, X., Huang, S. and Wan, Z. (2016) Optimal Pricing and Ordering in Global Supply Chain Management with Constraints under Random Demand. *Applied Mathematical Modelling*, **40**, 10105-10130. https://doi.org/10.1016/j.apm.2016.06.054

[13] Li, T. and Wan, Z. (2019) New Adaptive Barzilai-Borwein Step Size and Its Application in Solving Large-Scale Optimization Problems. T*he ANZIAM Journal*, **61**, 76-98. https://doi.org/10.1017/S1446181118000263

[14] 李化欣, 潘晋孝. 最速下降法在图像重建中的应用[J]. 科技情报开发与经济, 2006, 16(3): 155-156.

[15] Farsiu, S., Robinson, M.D., Elad, M. and Milanfar, P. (2004) Fast and Robust Multiframe Super Resolution. *IEEE Transactions on Image Processing*, **13**, 1327-1344. https://doi.org/10.1109/TIP.2004.834669