深度学习中优化器综述

来文博 *22210180088*　　龚畅 *22210980103*

ABSTRACT

深度学习中最常用的参数迭代算法莫过于随机梯度下降 (SGD)，在各种任务中都展现出其强大的通用性。近年来，基于 SGD 的优化器改进算法层出不穷，但很少有论文对它们在不同任务上的效果和效率进行全面的研究。本文综合考察了 8 种在深度学习中比较热门的优化器算法 (简称为一阶方法)。同时，考虑到二阶方法 (如牛顿法) 在传统优化领域的重要地位，本文也调研了其中的 5 种。通过将一、二阶方法以不同方式结合，本文提出了硬软两种融合方法。本文在 4 种数据集上对所有方法进行了全面的实验测试，得出了一阶方法普遍优于二阶方法的结论。同时，在数据规模较小时，也可适当考虑二阶方法。我们的代码开源在 https://github.com/hmtbgc/optimizer_survey

Index Terms— 优化器、一阶方法、二阶方法

1. 简述

近几年，深度学习在各领域大放异彩，其中随机梯度下降 (SGD) 作为最流行的参数更新方式被广泛应用于各种神经网络的训练，包括计算机视觉的 ResNet [1]、自然语言处理的 Transformer [2]、图表示的 GCN [3] 等。各大深度学习框架 (Tensorflow [4]、Pytorch [5]、Caffe [**?**] 等) 也都全面支持 SGD 训练，并实现了多种基于 SGD 改进的优化器算法。但是很少有论文分析这些一阶方法在不同任务上的实验结果，并针对具体问题给出合理建议。同时二阶方法 (如牛顿法) 在传统优化领域具有重要地位，将这一类优化器算法纳入比较也是值得考虑的问题。[6] 虽然较为全面地介绍了这些一阶方法，但仅在单个数据集 (Beale function) 上进行了测试，而且缺少对二阶方法的综述和实验。本文的主要贡献在于综述深度学习中常见的优化器和传统优化领域的二阶方法，并在多个数据集上进行了全面详细的测试，得出了较为丰富且泛化的实验结果，利于之后深度学习优化器研究的参考。

本文的结构如下：第 2 部分将详细介绍一二阶优化器算法，并选取若干算法进行理论分析；第 3 部分将所有方法应用于 4 个数据集，并报告实验结果；第 4 部分为总结，并对未来工作给出建议。

2. 优化算法主体及相关理论分析

第一部分分为一阶算法、二阶算法和一阶算法与二阶算法的融合三个方面介绍。在每一个算法中着重给出其最核心的部分：算法的迭代过程。

对于一个优化问题，设其目标函数为 $f(\theta)$，已知其当前迭代得到的参数集合为 θ_t，下列各算法给出了 θ_{t+1} 的迭代过程。

2.1. 一阶算法

本文主要介绍 SGD、带动量的 SGD、带 Nesterov 动量的 SGD、Adagrad、RMSprop、Adam、AdaMax、NAdam 等一阶算法。

2.1.1. SGD

SGD 是最简单的优化算法，其迭代方向为目标函数在 θ_t 的负梯度方向，迭代的数学表达如下：

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta f(\theta_t)$$

其中，α 是固定学习率. 对于该方法，α 的选取很关键: 如果 α 取值过小，算法会收敛很慢; 若取值过大则会导致目标函数振荡不收敛。在实验中，我们取 $\alpha \in \{0.01, 0.001, 0.005\}$.

下面考虑一般情形 SGD 算法的收敛性分析 [7]。一般情形下，考虑到抽取样本的不同，每一步的目标函数为 $f_t(\theta)$，SGD 变量的迭代方式为

$$\theta_{t+1} = \theta_t - \alpha_t g_t$$

其中 α_t 满足 $\alpha_1 \geq \alpha_2 \geq ... \geq \alpha_T \geq ... \geq 0$ 为学习率 (非固定)，而

$$g_t = \nabla_\theta f(\theta_t)$$

对于任意 t，$f_t(\theta)$ 都是关于 θ 的凸函数。判定指标选择为统计量 $R(T)$

$$R(T) = \sum_{t=1}^{T} f_t(\theta_t) - min_\theta \sum_{t=1}^{T} f_t(\theta)$$

若当 $T \to \infty$ 时，$R(T)/T \to 0$ 我们认为这时算法是收敛的，即

$$\theta \to argmin_\theta \sum_{t=1}^{T} f_t(\theta) \triangleq \theta^*$$

为了证明算法的收敛性，有两点假设：

1. 变量有界假设：

$$\|\theta - \theta'\|_2 \leq D, \forall \theta, \theta'$$

2. 梯度有界假设：

$$\|g_t\|_2 \leq G, \forall t$$

从统计量 $R(T)$ 开始，

$$\begin{aligned} R(T) &= \sum_{t=1}^{T} f_t(\theta_t) - min_\theta \sum_{t=1}^{T} f_t(\theta) \\ &= \sum_{t=1}^{T} f_t(\theta_t) - \sum_{t=1}^{T} f_t(\theta^*) \\ &= \sum_{t=1}^{T} \Big[f_t(\theta_t) - f_t(\theta^*) \Big] \end{aligned}$$

由于 $f_t(\theta)$ 是凸函数，有

$$f_t(\theta_t) - f_t(\theta^*) \leq \langle g_t, \theta_t - \theta^* \rangle$$

故

$$R(T) \leq \sum_{t=1}^{T} \langle g_t, \theta_t - \theta^* \rangle$$

从 θ_t 的迭代式开始做变形：

$$\begin{aligned} &\theta_{t+1} = \theta_t - \alpha_t g_t \\ \Rightarrow &\theta_{t+1} - \theta^* = \theta_t - \theta^* - \alpha_t g_t \\ \Rightarrow &\|\theta_{t+1} - \theta^*\|_2^2 = \|\theta_t - \theta^* - \alpha_t g_t\|_2^2 \\ \Rightarrow &\|\theta_{t+1} - \theta^*\|_2^2 = \|\theta_t - \theta^*\|_2^2 - 2\alpha_t \langle g_t, \theta_t - \theta^* \rangle \\ &\quad + \alpha_t^2 \|g_t\|_2^2 \\ \Rightarrow &\langle g_t, \theta_t - \theta^* \rangle = \frac{1}{2\alpha_t} \Big[\|\theta_t - \theta^*\|_2^2 - \|\theta_{t+1} - \theta^*\|_2^2 \Big] \\ &\quad + \frac{\alpha_t}{2} \|g_t\|_2^2 \end{aligned}$$

接着整理出 $R(T)$ 的上界

$$\begin{aligned} R(T) \leq &\sum_{t=1}^{T} \frac{1}{2\alpha_t} \Big[\|\theta_t - \theta^*\|_2^2 - \|\theta_{t+1} - \theta^*\|_2^2 \Big] \quad (1) \\ &+ \sum_{t=1}^{T} \frac{\alpha_t}{2} \|g_t\|_2^2 \quad (2) \end{aligned}$$

下面对 $R(T)$ 的上界进行放缩，首先看 (2) 式，根据梯度有界假设

$$\sum_{t=1}^{T} \frac{\alpha_t}{2} \|g_t\|_2^2 \leq \sum_{t=1}^{T} \frac{\alpha_t}{2} G^2 = \frac{G^2}{2} \sum_{i=1}^{T} \alpha_t$$

接着我们看 (1) 式，将 (1) 逐项展开得到

$$\begin{aligned} &\sum_{t=1}^{T} \frac{1}{2\alpha_t} \Big[\|\theta_t - \theta^*\|_2^2 - \|\theta_{t+1} - \theta^*\|_2^2 \Big] \\ =&\frac{1}{2\alpha_1} \|\theta_1 - \theta^*\|_2^2 + \sum_{t=2}^{T} \Big(\frac{1}{2\alpha_t} - \frac{1}{2\alpha_{t-1}} \Big) \|\theta_t - \theta^*\|_2^2 \\ &- \frac{1}{2\alpha_T} \|\theta_{T+1} - \theta^*\|_2^2 \end{aligned}$$

- 根据变量有界假设，$\frac{1}{2\alpha_1} \|\theta_1 - \theta^*\|_2^2 \leq \frac{1}{2\alpha_1} D^2$
- 由于 $\{\alpha_t\}$ 单调不增，再结合变量有界假设，则

$$\sum_{t=2}^{T} \Big(\frac{1}{2\alpha_t} - \frac{1}{2\alpha_{t-1}} \Big) \|\theta_t - \theta^*\|_2^2 \leq D^2 \sum_{t=2}^{T} \Big(\frac{1}{2\alpha_t} - \frac{1}{2\alpha_{t-1}} \Big)$$

故 (1) 最终可放缩为

$$\begin{aligned} &\sum_{t=1}^{T} \frac{1}{2\alpha_t} \Big[\|\theta_t - \theta^*\|_2^2 - \|\theta_{t+1} - \theta^*\|_2^2 \Big] \\ \leq &\frac{1}{2\alpha_1} D^2 + D^2 \sum_{t=2}^{T} \Big(\frac{1}{2\alpha_t} - \frac{1}{2\alpha_{t-1}} \Big) = D^2 \frac{1}{2\alpha_T} \end{aligned}$$

最终整理出 $R(T)$ 的上界为

$$R(T) \leq D^2 \frac{1}{2\alpha_T} + \frac{G^2}{2} \sum_{t=1}^{T} \alpha_t$$

下面考虑最优学习率的设计，令 α_t 是 t 的函数，且采用多项式衰减：$\alpha_t = C/t^p$, $p \geq 0$

$$\begin{aligned} R(T) &\leq D^2 \frac{T^p}{2C} + \frac{G^2}{2} \sum_{t=1}^{T} \frac{C}{t^p} \\ &\leq D^2 \frac{T^p}{2C} + \frac{G^2 C}{2} \Big(1 + \int_1^T \frac{dt}{t^p}\Big) \\ &= D^2 \frac{T^p}{2C} + \frac{G^2 C}{2} \Big(\frac{1}{1-p} T^{1-p} - \frac{p}{1-p}\Big) \end{aligned}$$

于是 $R(T) = O\big(T^{max(p,1-p)}\big)$. 当 $p = 1/2$ 时，$R(T)$ 取得最优上界 $O\big(T^{1/2}\big)$，相应的学习率 $\alpha_t = C/t^{1/2}$，均摊值 $R(T)/T = O\big(T^{-1/2}\big)$ 在 $T \to \infty$ 时趋于 0.

2.1.2. 带动量的 SGD

第一个对于 SGD 的改进方法就是基于它一个动量 [**?**]，可看作是一个球滚下一个恒定斜坡时所具备的潜在的能量，体现在迭代过程中如下：

$$m_t = \gamma m_{t-1} + \alpha \nabla_\theta f(\theta_t)$$

$$\theta_{t+1} = \theta_t - m_t$$

其中 γ 为一恒定的常数，通常取为 0.9，α 为学习率。

2.1.3. 带 Nesterov 动量的 SGD

带 Nesterov 动量的 SGD 是对带动量的 SGD 的改进 [8]，在动量更新的式子中不再是加上 θ_t 处的梯度，而是 $\theta_t - \gamma m_{t-1}$ 处的梯度，其迭代式为：

$$m_t = \gamma m_{t-1} + \alpha \nabla_\theta f(\theta_t - \gamma m_{t-1})$$

$$\theta_{t+1} = \theta_t - m_t$$

如下两张图直观地展示了带动量的 SGD 与带 Nesterov 动量的 SGD 的区别：
如图 2 所示，当此刻的动量很大时，带 Nesterov 动量的 SGD 则会对此刻的迭代进行适当的调整。

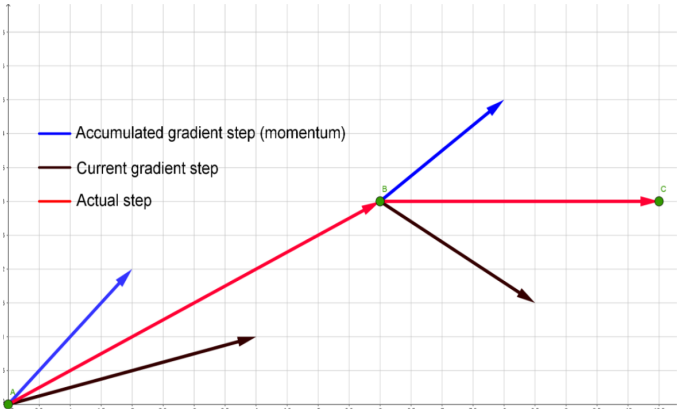

Fig. 1. SGD+Momentum

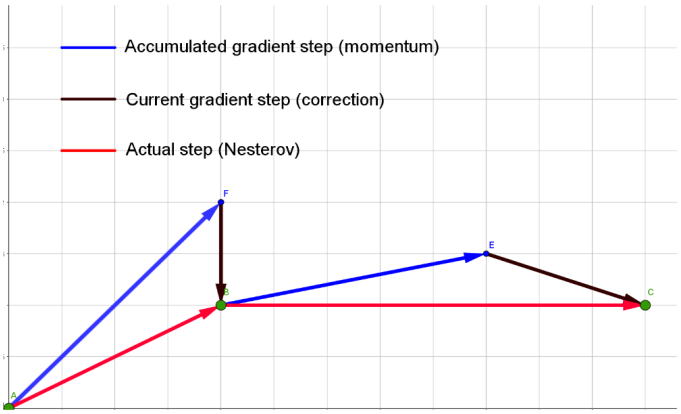

Fig. 2. SGD+Momentum

2.1.4. AdaGrad

Adagrad [9] 的想法是使得更新频率与学习率成负相关，因此在迭代过程中学习率的分母上赋予当前全部梯度的 L^2 范数的形式，具体迭代过程如下：

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\sum_{T=1}^{t} g_T^2 + \epsilon}} g_t$$

其中 ϵ 为保险系数，通常取为 10^{-8}。

2.1.5. RMSprop

相比于 Adagrad 中学习率递减，有可能很快变为 0，RMSprop [10] 对算法做了调整，通过构造 v_t 并通过指数移动平均公式来更新它，从而取代了梯度的累积。其算法迭代过程如下：

$$v_t = \gamma v_{t-1} + (1-\gamma) g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{v_t + \epsilon}} g_t$$

2.1.6. Adam

Adam [11] 将 Adagrad 和 RMSprop 融合，算法中定义了两个动量 m_t 和 v_t，其更新表达式如下：

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

通常情况下 m_t 和 v_t 的迭代初值均为 0，导致在初期动量对参数 θ 的更新贡献很小，因此对 m_t 和 v_t 作了如下修正：

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

参数 θ 的更新过程为：

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

也可以写作：

$$\theta_{t+1} = \theta_t - \alpha \frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t} \frac{m_t}{\sqrt{v_t} + \epsilon}$$

在后续的算法实现中，β_1 取值为 0.9，β_2 取值为 0.999。

对于 Adam 算法的收敛性分析，我们此处只介绍如下定理 [11]。我们沿用 SGD 中收敛性的判别统计量 $R(T)/T$（其中 $R(T) = \sum_{t=1}^{T} \left[f_t(\theta_t) - f_t(\theta^*) \right]$），我们认为若当 $T \to \infty$ 时，$R(T)/T \to 0$ 则算法收敛。首先定义如下符号:

$$g_{1:t,i} \triangleq [\, g_{1,i}, g_{2,i}, ..., g_{t,i} \,], \quad \gamma \triangleq \frac{\beta_1^2}{\sqrt{\beta_2}}$$

该算法收敛结果由如下定理给出：

定理 设目标函数 f_t 具有有界的梯度，即 $\|\nabla f_t(\theta)\|_2 \le G$，$\|\nabla f_t(\theta)\|_\infty \le G_\infty$ 且变量有界，即对 $\forall \theta_m, \theta_n \in \{1, 2, ..., T\}$，有 $\|\theta_m - \theta_n\|_2 \le G$，且 $\beta_1, \beta_2 \in [0, 1)$，如上定义的 $\gamma < 1$. 令 $\alpha_t = \frac{\alpha}{\sqrt{t}}$ 且 $\beta_{1,t} = \beta_1 \lambda^{t-1}$ 则对 $T \ge 1$ 有如下结果：

$$\begin{aligned} R(T) \le & \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} \sqrt{T \hat{v}_{T,i}} \\ & + \frac{\alpha(1+\beta_1) G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} \|g_{1:T,i}\|_2 \\ & + \sum_{i=1}^{d} \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha(1-\beta_1)(1-\lambda)^2} \end{aligned}$$

该定理的证明见参考文献 [11]。

在满足以上条件的同时，我们有 $\sum_{i=1}^{d} \|g_{1:T,i}\|_2 << dG_\infty \sqrt{T}$ 以及 $\sum_{i=1}^{d} \sqrt{T \hat{v}_{T,i}} << dG_\infty \sqrt{T}$ 再结合上述定理，有如下结果：

$$\frac{R(T)}{T} = O(\frac{1}{\sqrt{T}})$$

故当 $T \to \infty$ 时，$R(T)/T \to 0$ 则算法收敛。

2.1.7. AdaMax

Adam 算法中 v_t 的更新过程可改为 L^p 范数的形式如下：

$$v_t = \beta_2^p v_{t-1} + (1 - \beta_2^p) \|g_t\|_p^p$$

随着 p 的增大，计算过程越来越不稳定；若上述过程中取无穷范数，则得到 AdaMax 算法 [11]：

$$v_t = max(\beta_2 v_{t-1}, |g_t|)$$

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{v_t + \epsilon}$$

其中 $\hat{m}_t$ 及 α 和 ϵ 取值与 Adam 算法相同。此外，该方法在运算过程中除去了内积和开根号的运算，使得运行时间得以减少。

2.1.8. NAdam

类似地，可在 Adam 算法中的 m_t 更新过程中加以变化，从而得到新的 NAdam 算法 [12]：

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} = \frac{\beta_1 m_{t-1} + (1 - \beta_1) g_t}{1 - \beta_1^t}$$

可以引入新的变化量 $\overline{m}_t$:

$$\overline{m}_t = \frac{\beta_1 \hat{m}_t + (1 - \beta_1) g_t}{1 - \beta_1^t}$$

则 θ 的更新公式为：

$$\theta_{t+1} = \theta_t - \alpha \frac{\overline{m}_t}{\sqrt{v_t} + \epsilon}$$

2.2. 二阶算法

二阶算法在迭代过程中多了计算函数 Hessian 矩阵的运算量，在运算时间和收敛性上面对算法具有了更高的要求。本文主要介绍 Newton 法、BFGS 算法、L-BFGS 算法、Hessian-Free 算法、带动量的 BFGS（L-BFGS）算法等方法。

2.2.1. Newton

将目标函数进行二阶泰勒展开：

$$f(\theta+\Delta\theta)=f(\theta)+\nabla f(\theta)^T\Delta\theta+\frac{1}{2}\Delta\theta^T H(\theta)\Delta\theta$$

上式右端对 $\Delta\theta$ 求梯度并令其等于 0 得到 Newton 法的迭代方向，其迭代过程为：

$$\theta_{t+1}=\theta_t-\alpha\, H(\theta_t)^{-1}\nabla f(\theta_t)$$

该方法需要求 Hessian 矩阵和它的逆矩阵，考虑到逆矩阵计算的复杂性，该算法在时间和空间上开销巨大。

2.2.2. BFGS

由于 Newton 法中 Hessian 矩阵往往不易求得，故一些拟牛顿法中用一些满足一定迭代式的矩阵来代替 Heesian 矩阵，BFGS 算法就是其一。在 BFGS 算法中，迭代时刻 t 式的 Hessian 矩阵的替代矩阵为 B_t，相应的其逆矩阵为 H_t，矩阵 H_t 所满足的迭代式为：

$$\begin{aligned}H_t=&(I-\rho_{t-1}s_{t-1}y_{t-1}^T)H_{t-1}(I-\rho_{t-1}y_{t-1}s_{t-1}^T)\\&+\rho_{t-1}s_{t-1}s_{t-1}^T\end{aligned}$$

其中 $\rho_{t-1}=1/y_{t-1}^T s_{t-1}$，

$$s_{t-1}=\theta_t-\theta_{t-1},\ y_{t-1}=\nabla f(\theta_t)-\nabla f(\theta_{t-1})$$

θ 的迭代过程为：

$$\theta_{t+1}=\theta_t-\alpha\, H_t\nabla f(\theta_t)$$

2.2.3. L-BFGS

在 BFGS 算法中，H_t 可用 H_{t-1} 表示，同样 H_{t-1} 可用 H_{t-2} 表示，此表达式可代入 H_t 的迭代式中；如此递归迭代 m 次即为 L-BFGS 算法的思想，具体迭代过程如下：

$$\begin{aligned}H_t=&(V_{t-1}^T\cdots V_{t-m}^T)H_t^0(V_{t-1}\cdots V_{t-m})\\&+\rho_{t-m}(V_{t-1}^T\cdots V_{t-m+1}^T)s_{t-m}s_{t-m}^T(V_{t-1}\cdots V_{t-m+1})\\&+\rho_{t-m+1}(V_{t-1}^T\cdots V_{t-m+2}^T)s_{t-m+1}s_{t-m+1}^T\\&(V_{t-1}\cdots V_{t-m+2})+\rho_{t-1}s_{t-1}s_{t-1}^T\end{aligned}$$

$$s_j=\theta_{j+1}-\theta_j\quad y_j=\nabla f(\theta_{j+1})-\nabla f(\theta_j)$$

$$\rho_j=1/y_j^T s_j\quad V_j=I-y_j s_j^T$$

θ 的迭代过程与 BFGS 算法一致：

$$\theta_{t+1}=\theta_t-\alpha\, H_t\nabla f(\theta_t)$$

2.2.4. Hessian-Free

Hessian-Free [13] 是另外一种可以避免求 Hessian 矩阵及其逆的方法，其过程如下：
由 Newton 法，在 t 时刻需要求解方程

$$H(\theta_t)p_t=-\nabla f(\theta_t)$$

用 Newton-CG 算法求解，迭代第 t 步的搜索方向满足

$$d_t=-\nabla f(\theta_{t-1})+\frac{\nabla f(\theta_t)H(\theta_{t-1})d_{t-1}}{d_{t-1}^T H(\theta_{t-1})d_{t-1}}d_{t-1}$$

其中，可用 Hessian-Vector-Product 方法求 Hd:

$$Hd=lim_{\epsilon\to 0}\frac{\nabla f(\theta+\epsilon d)-\nabla f(\theta)}{\epsilon}$$

θ 的迭代过程为：

$$\theta_{t+1}=\theta_t+\alpha\ p_t$$

该算法所带来的问题则是每次 CG 都需要迭代一定的步骤才能收敛到解，比较耗时；且 Hessian-Vector-Product 方法对目标函数的可微性及方向导数要求较高。

2.2.5. 带动量的 BFGS(L-BFGS)

仿照带动量的 SGD 方法，将动量的思想代入 BFGS、L-BFGS [14]，新算法迭代过程如下：

设时刻 t 由 BFGS 算法（L-BFGS）算法所生成的方向为 p_t，引入动量 m_t：

$$m_t = \gamma m_{t-1} + \alpha\, p_t$$

$$\theta_{t+1} = \theta_t - m_t$$

后续算法实现中取 $\gamma = 0.9$。

2.3. 一阶算法与二阶算法的融合

2.3.1. 一阶方向和二阶方向的硬结合

设时刻 t 某一个一阶算法得到的更新方向为 c_t，某一个二阶方法得到的更新方向为 p_t，则将二者作算术平均得到此时的更新方向，迭代过程为：

$$b_t = \gamma c_t + (1-\gamma) p_t$$

$$\theta_{t+1} = \theta_t - \alpha\, b_t$$

后续算法实现中取 $\gamma = 0.9$。

2.3.2. 一阶方向和二阶方向的软结合

硬结合每次迭代都需要计算一阶方法和二阶方法的更新方向，可能造成单次迭代时间开销过大。另一方面，难以设置硬结合中的超参 γ。考虑到二阶方法比较耗时，但收敛效果较好，可以在迭代初期使用，并逐步过渡到一阶方法，确保收敛。因此我们采用软分割的方式：

$$\beta(t) = 1 - e^{-\alpha t}, \alpha > 0$$

$$b_t = \beta(t) c_t + (1-\beta(t)) p_t$$

且当 $\beta(t) > threshold$ 时，不计算二阶方法的更新方向来减少时间开销。

3. 优化算法的具体实现

第二部分我们通过几个具体的例子对上述优化算法进行了实践，并比较各算法在各个具体的问题中应用的效果。

3.1. 马鞍面的鞍点逃离问题

首先考虑马鞍面的鞍点逃离问题，定义马鞍面的函数表示为：

$$z = x^2 - y^2$$

绘制该图形如下：

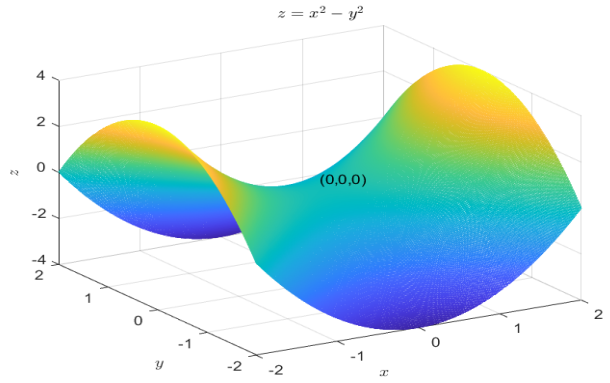

Fig. 3. 马鞍面图像

如图 3 所示，$(0,0,0)$ 为其一个鞍点；我们应用上述算法探究赋予该鞍点附近的初值时各算法迭代是否会逃离鞍点。

我们将上述各算法均迭代 2000 步，并比较各算法的迭代结果。各算法的鞍点的逃离情况用误差和距离随时间的变化来衡量；其中，误差为 z 的值，距离为点 (x,y,z) 与鞍点 $(x_0,y_0,z_0) = (0,0,0)$ 之差的二范数表示，即：

$$dist = \|(x,y,z) - (x_0,y_0,z_0)\|_2$$

首先我们给定各算法的迭代初值为 $(1,0)$，绘制各一阶算法和二阶算法距离随时间的变化图像如下：

如图 4、图 5 所示，在给定初值为 (1,0) 时，一阶算法和二阶算法均有向鞍点靠拢的趋势，且其中带动量的 SGD 算法、带 Nesterov 动量的 SGD 算法、RMSprop 算法、Adamax 算法、NAdam 算法以及带动量的 BFGS 算法均在数百步收敛至鞍点并停留至鞍点。

考虑到初值为 (1,0) 使得各算法的迭代过程基本位于二维曲线 $z = x^2$ 上，变为一凸函数的优化问题，

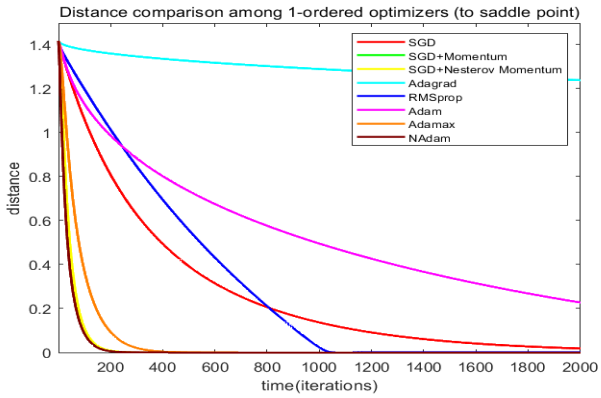

Fig. 4. 马鞍面初值 (1,0) 的一阶算法误差

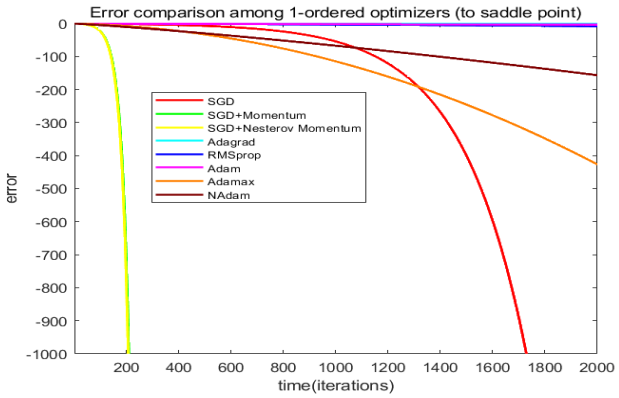

Fig. 6. 马鞍面初值 (1,1) 的一阶算法误差/大比例尺

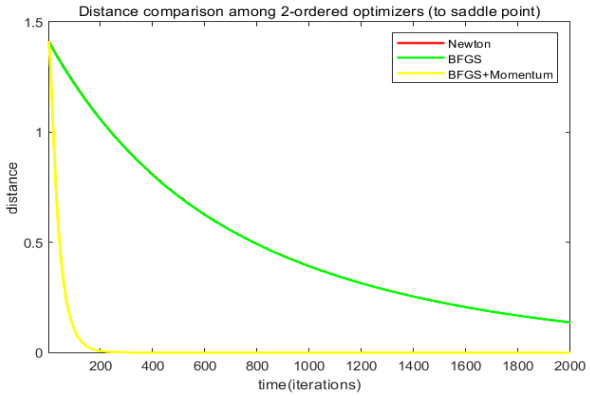

Fig. 5. 马鞍面初值 (1,0) 的一阶算法距离

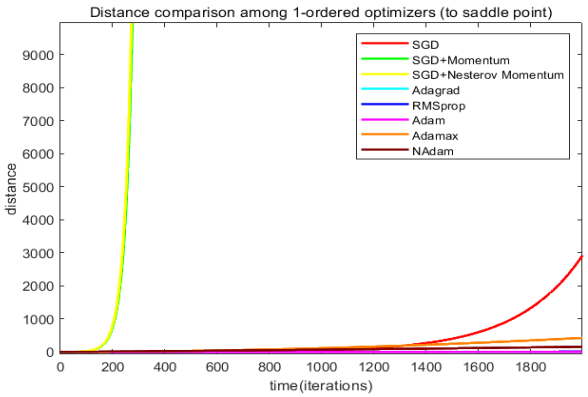

Fig. 7. 马鞍面初值 (1,1) 的一阶算法距离/大比例尺

难免会导致算法困于鞍点，故我们另取一组初值 (1,1)，得到的各算法迭代结果如下，首先是一阶算法：

如图 6、图 7 所示，一阶算法中，相比于其他算法，SGD 算法以及带动量的 SGD 算法、带 Nesterov 动量的 SGD 算法能够迅速逃离鞍点并使得 z 的值趋于 $-\infty$. 其中，又以带动量的 SGD 算法和带 Nesterov 动量的 SGD 算法逃离速度最快。为比较其他一阶算法的逃离情况，我们缩小比例尺绘制了图 8、图 9 如下：

如图 8，图 9 所示，Adamax 算法和 NAdam 算法逃离鞍点的速度远大于其他三个算法；而相反 AdaGrad 算法在 2000 步的迭代步骤中持续向鞍点靠近；Adam 算法以稳定的速率逃离鞍点；而 RMSprop 算法再经过更多的迭代步骤后，其逃离速率将远超过 Adam 算法。

接着是二阶算法在逃离鞍点中的效果：

如图 10，图 11 所示，BFGS 算法由于是基于 Newton 法的思想，故其与 Newton 法的迭代过程相似，均不能有效的逃离鞍点；而带动量的 BFGS 算法则能以显著的速度逃离鞍点。

综合一阶算法的过程，引入动量的思想对马鞍面的逃离鞍点的过程有着极大的加速作用。

最后是综合一阶算法和二阶算法的作用，这里选了一阶算法中的 NAdam 算法和二阶算法中的 Newton 法进行代数结合，算法迭代结果如下：

如图 12，图 13，受 Newton 法的影响，结合后的 Newton+NAdam 算法的逃离鞍点速率有所降低，且相比于 NAdam 算法其逃离速率趋于平稳。

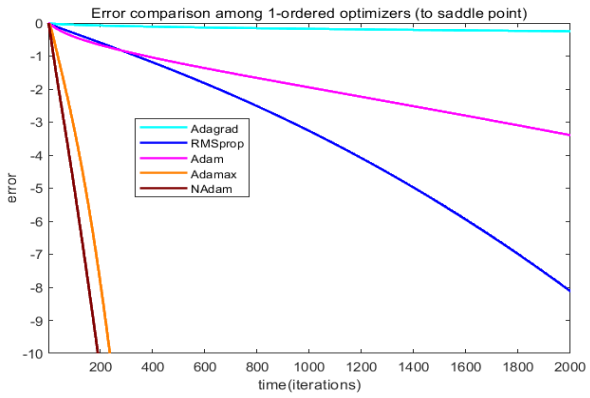

Fig. 8. 马鞍面初值 (1,1) 的一阶算法误差/小比例尺

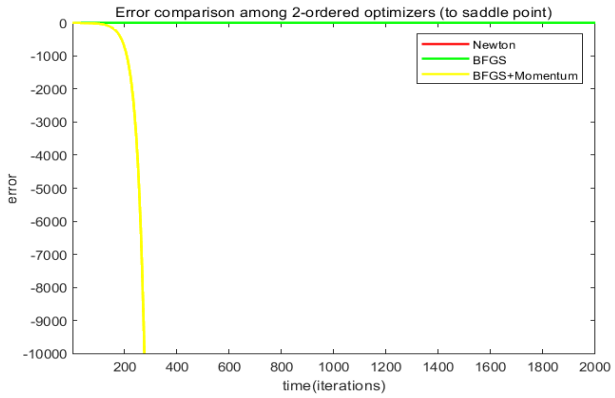

Fig. 10. 马鞍面初值 (1,1) 的二阶算法误差

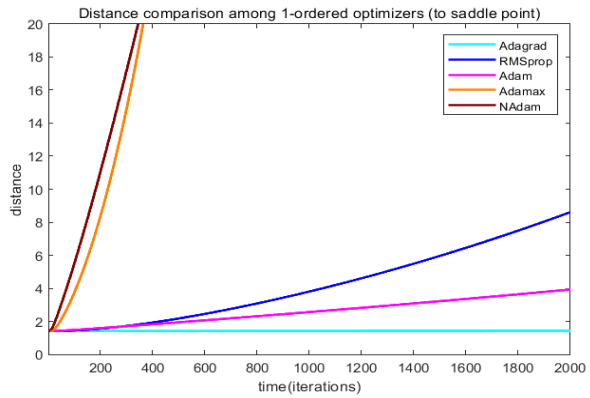

Fig. 9. 马鞍面初值 (1,1) 的一阶算法距离/小比例尺

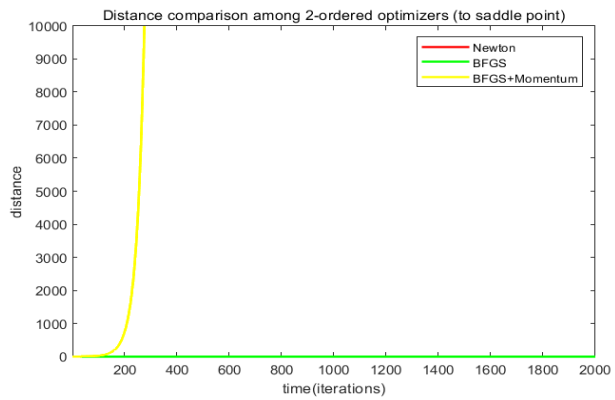

Fig. 11. 马鞍面初值 (1,1) 的二阶算法距离

3.2. 自造函数的优化问题

第二部分算法的实践我们考虑一个自造函数的优化问题，定义这个二元函数有以下表达式：

$$f(x,y)=\frac{ln\Big[1+(1.5+x-xy)^2+(2.25+x-xy^2)^2}{10}\\ \frac{+(2.625-x+xy^3)^2\Big]}{10}$$

绘制该二元函数表达式如下：

如图 14，该目标函数在点 $(3,0.5)$ 取得最小值 0. 我们应用各算法对该函数进行求最小值并比较各算法的收敛情况。

在后续的算法实现中，取迭代初值为 $(2,0)$，迭代步数为 2000 步。类似 2.1 马鞍面问题的讨论，我们定义误差为迭代过程中函数 $f(x,y)$ 的值，定义距离为 R^3 中的点 $(x,y,f(x,y))$ 与目标点 $(3,0.5,0)$ 之差的二范数。

下面是各算法的迭代结果，首先是一阶算法，一阶算法的迭代结果：

如图 15，图 16，SGD 算法与 AdaGrad 算法收敛效果相对较差，但 SGD 算法趋于最优解的速度要大于 AdaGrad 算法；而 Adam 算法和带动量的 SGD 算法以及带 Nesterov 动量的 SGD 算法趋于最优解的速度相当，但在 2000 步的迭代中未收敛至最优解；此外 NAdam 算法、Adamax 算法、RMSprop 算法分别在大约 800 步、1000 步、1300 步时收敛至最优解。

接着是二阶算法在目标函数优化中的效果：

如图 17, 图 18 所示, 同样由于 BFGS 算法思想源于 Newton 法, 故这两个算法迭代过程基本相似, 且加

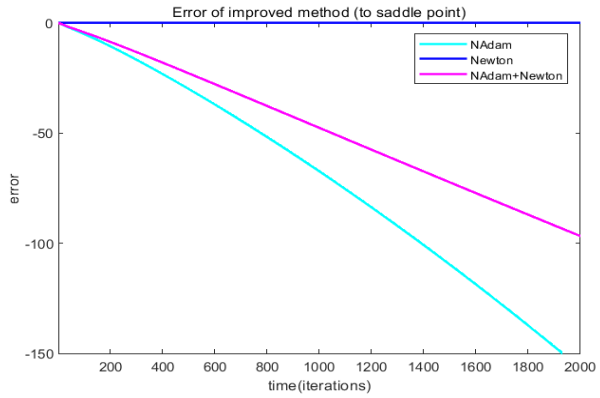

Fig. 12. 马鞍面初值 (1,1) 的一阶结合二阶算法误差

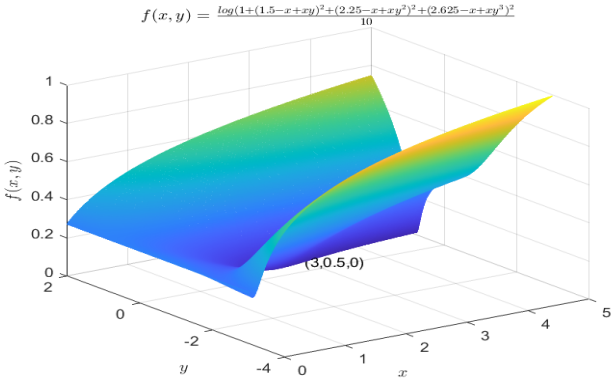

Fig. 14. 自造目标函数图像

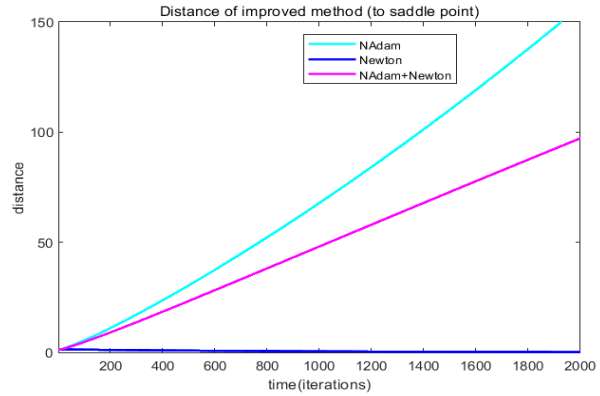

Fig. 13. 马鞍面初值 (1,1) 的一阶结合二阶算法距离

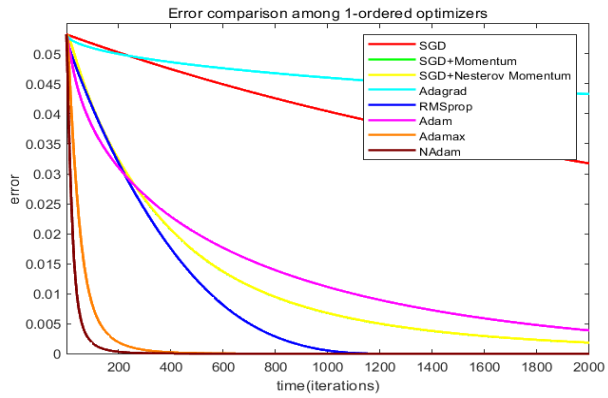

Fig. 15. 一阶算法的目标函数误差

上动量并未使得 BFGS 算法的收敛速度有明显提升。综合来看，Newton 法、BFGS 算法和带动量的 BFGS 算法经过 2000 步的迭代后，目标函数的值与带动量的 SGD 算法和带 Nesterov 动量的 SGD 算法相近。

此外，在二阶算法中，Hessian-Free 算法的收敛效果奇差，可能是因为被优化目标函数方向导数性质不稳定，Hessian-Vector-Product 算法中的极限不一定存在，导致 Hessian-Free 算法不具有好的收敛性。该算法迭代结果如图 19，图 20 所示:

最后来看一阶算法与二阶算法结合的收敛情况，我们同样将一阶算法中的 NAdam 算法与二阶算法中的 Newton 法作代数结合；这里只展示结合后算法的目标距离随时间的变化情况如下:

如图 21 所示，由于受到 Newton 法的影响，在前期结合后的算法收敛速度要慢于原 NAdam 算法，但最终都在第 800 步左右收敛至最优解。

综上所述，在该目标函数的优化问题中，NAdam 算法、Adam 算法及 RMSprop 算法收敛效果较好，均能在 2000 步之内收敛至最优解。同时，二阶算法的收敛效果整体不如一阶算法；此外，将二者结合后，二阶算法并未对原一阶算法的收敛起到加速作用。

3.3. 二分类问题

该节分析优化器在小型数据集上的结果。我们选取的数据集出自 CS229 Assignment 2，共 100 行，每一行有 3 个实数。前两个对应学生的两门功课分数，最后一个表示学生是否能考上大学：0 表示不能，1 表示可以。该二分类任务目的在于依据学生两门功课成

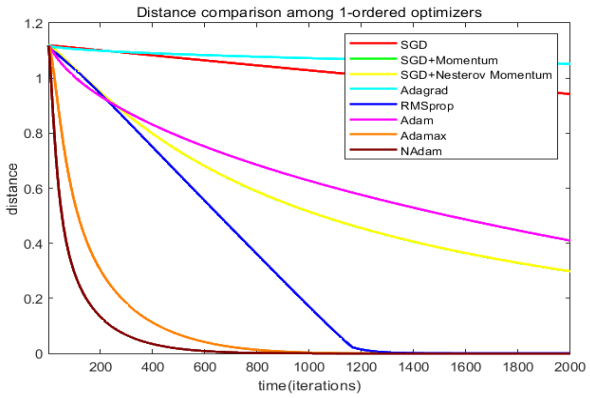

Fig. 16. 一阶算法的目标点距离

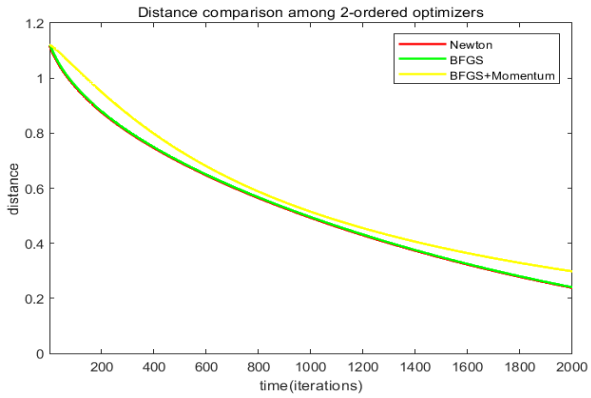

Fig. 18. 二阶算法的目标点距离

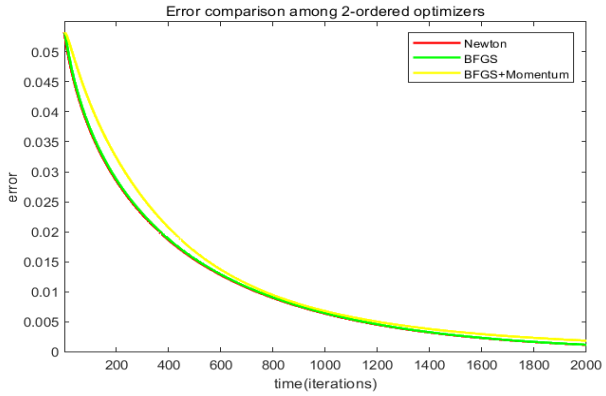

Fig. 17. 二阶算法的目标函数误差

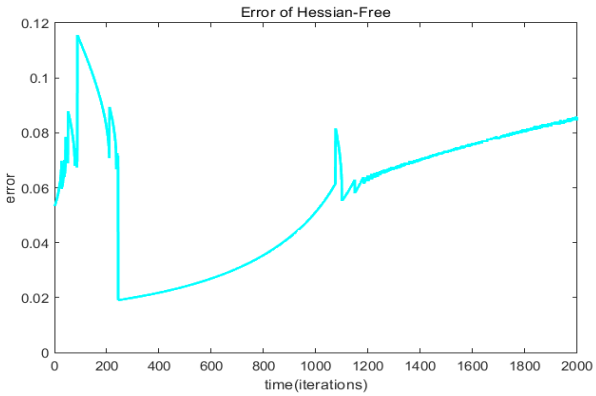

Fig. 19. Hessian-Free 算法的目标函数误差

绩来预测其能否被大学录取。

对于二分类问题，我们可以使用逻辑回归来建模：设第 i 个实例特征为 $x^{(i)} \in \mathbb{R}^2$，参数为 $\theta \in \mathbb{R}^2$，令 $h_\theta(x) = g(\theta^T x)$，其中 $g(z) = \frac{1}{1+e^{-z}}$，则损失函数可以表示为：

$$l(\theta) = -\log \Pi_{i=1}^{n} h_\theta(x^{(i)})^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}}$$

$$= \sum_{i=1}^{n} -y^{(i)} \log h_\theta(x^{(i)}) - (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))$$

由于 $g'(z) = g(z)(1 - g(z))$，可以手算 $\frac{\partial l(\theta)}{\partial \theta_j} (j = 1, 2)$：

$$\frac{\partial l(\theta)}{\partial \theta_j} = \sum_{i=1}^{n} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

进一步可以计算 $\frac{\partial^2 l(\theta)}{\partial \theta_j \partial \theta_k} (j, k = 1, 2)$：

$$\frac{\partial^2 l(\theta)}{\partial \theta_j \partial \theta_k} = \sum_{i=1}^{n} h_\theta(x^{(i)})(1 - h_\theta(x^{(i)})) x_j^{(i)} x_k^{(i)}$$

我们选取所有的一阶方法，以及 Newton(CG)、Hessian-free、BFGS、BFGS-momentum 等二阶方法，主要比较以下几点：

1. 验证集精度变化趋势
2. 测试集精度

3.3.1. 验证集精度变化趋势

针对一阶方法，我们迭代 100000 次，并每 100 次迭代都计算一次验证集上的精度。针对二阶方法，我们采用回溯线搜索来寻找迭代的步长，并在牛顿增量的模长小于 ϵ 时提前终止迭代。实验结果如图 22、23、24 所示。

综合三张图可得，一阶方法普遍表现更好。在训练前期，验证集精度随着迭代快速上升，至少达到 87.5% 左右。而在迭代的后期，虽然会出现精度反复横跳的

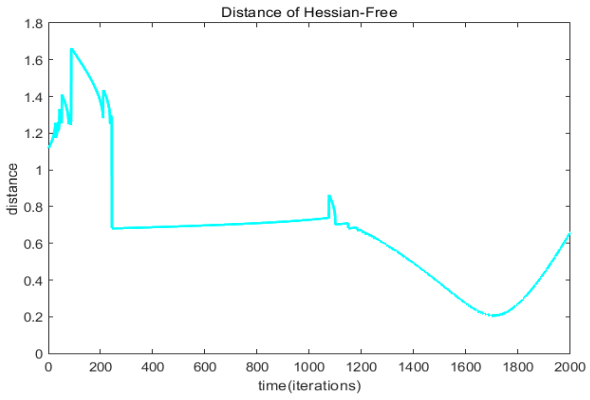

Fig. 20. Hessian-Free 算法的目标点距离

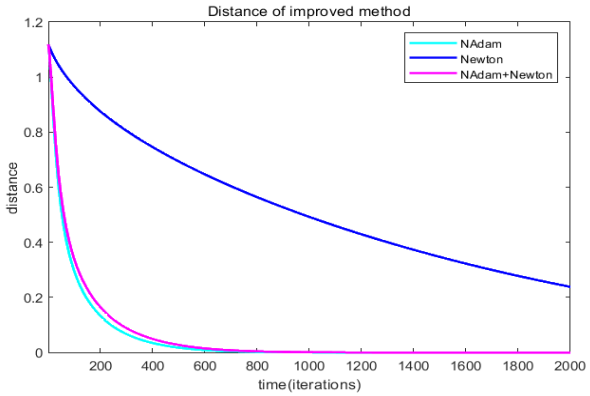

Fig. 21. 一阶结合二阶算法的目标点距离

情况，但考虑到验证集本身较小，精度的变化意味着少量实例被错误分类 (一般为 1-2 个)，这是可以被接受的。在所有一阶方法中，SGD 作为最原始的 baseline，确实在收敛速度上要远逊于其各种改进。相比之下，二阶方法中的 Newton 和 Hessian-free 仅花了不足 10 步就结束迭代，验证集的精度也令人满意。而 BFGS 和其动量变体似乎表现不佳。前者的精度随着迭代出现大幅度的跳跃，且在迭代结束时依然难以收敛到稳定值；后者的精度从训练开始就保持在 60% 左右，且随着迭代进行没有任何改进。通过阅读原始论文，得知 BFGS-momentum 算法对学习率等超参的设置极其敏感，对于不同问题需要花费大量时间调参，因此随机设置的参数难以训练出满意的结果。

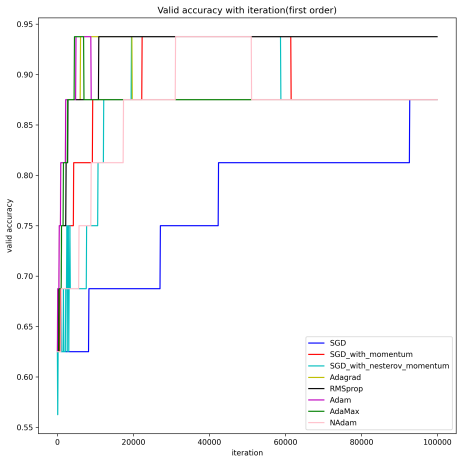

Fig. 22. 一阶方法验证精度变化趋势

3.3.2. 测试集精度

相比于损失函数和验证集精度变化，我们最关心的是模型在测试集上的精度高低。具体结果如表1所示。总体来说，一阶方法优于二阶方法。一阶方法中基于 SGD 改进的 Adam 类算法表现最好，二阶方法中 Newton 和 Hessian-free 又快又好。这符合如今深度学习界优化器选择的主流：自适应的 Adam 或带动量的 SGD。同时，如果问题规模不大，或者可以在合理时间内计算出 Hessian 矩阵及其逆，那 Newton 或 Hessian-free 也是不错的选择。

3.4. 多分类问题

在这小节中，我们关注多分类问题。我们选择的数据集是 Iris plants dataset [15], 来自 R.A. Fisher 的论文。数据集包含 150 个实例，每个实例有 4 个特征，总共分成 3 类。我们的目标是依据特征建立模型并对新的实例进行分类。

按照机器学习惯例，我们对 150 个实例进行训练集、验证集、测试集 6:2:2 的划分。我们的模型将在训练集上训练，在验证集上选取最优参数，最终在测试集上测试精度。由于数据集本身不复杂，我们采用线性模型：设训练集全体为 $X \in \mathbb{R}^{n\times 4}$，参数为 $A \in \mathbb{R}^{4\times 3}$，

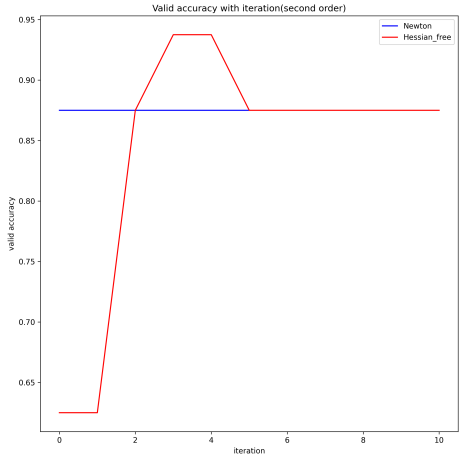

Fig. 23. 二阶方法验证精度变化趋势(Newton&Hessian-free)

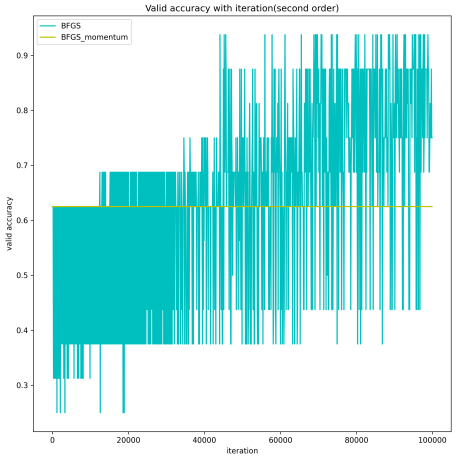

Fig. 24. 二阶方法验证精度变化趋势(BFGS(momentum))

训练标签为 $y \in \{0,1,2\}^n$, 则损失函数为:

$$\begin{aligned} l(A) &= cross\_entropy(XA, y) \\ &= -\sum_{i=1}^{n}(XA)_{(i,y_i)} + \sum_{i=1}^{n}\log\sum_{j=0}^{2}e^{(XA)_{(i,j)}} \end{aligned}$$

可以看到，手动求 $\frac{\partial l(A)}{\partial A}$ 是比较复杂的，进一步计算 Hessian 矩阵仅存在理论上的可能性。因此我们调用 autograd [16] 库的 grad 函数求梯度，用 hessian 函数求 Hessian 矩阵 (Hessian 仅为 12×12 的矩阵，在计算可承受范围内)。

类同上一节，我们主要考虑验证集上精度的趋势以及测试集上最终精度。为了更充分的分析，我们加入对训练集上损失函数变化趋势的考虑。

3.4.1. 损失函数变化趋势

我们首先考虑训练集上损失函数的变化趋势。一个有效的优化器应快速且稳定地对损失函数进行下降，且尽量避免损失函数大幅度振荡的情况，确保参数的变化有利于损失函数逐步变小。

图 25、26 分别展示了一、二阶方法的损失函数变化趋势。显然，所有方法的损失函数都呈下降趋势，区别在于下降速度的快慢。一阶方法中，带动量和带 Nesterov 动量的 SGD 下降速率最快，NAdam 表现得最糟糕。SGD 作为最原始的更新方式，在下降速率方面和 Adam、RMSprop、Adagrad 相差不多，且显著优于最晚提出的 AdaMax 和 NAdam。这表明近几年来虽然基于 SGD 提出了很多新的一阶方法，但 SGD 依然可以作为比较强劲的 baseline。在大部分情况下，使用 SGD 或其动量版本已经可以稳定获得较快的下降速度。对于二阶方法，动量加成的 BFGS 显著优于其他方法。注意到，Newton(CG) 和 Hessian-free 的曲线出现了重合，原因在于这两种算法都基于共轭梯度法 (CG) 的框架，区别仅仅是 CG 中 hessian-vector-product(hvp) 的计算方式不同。而这个问题规模较小，Hessian 矩阵不大，因此采用 Newton(CG) 的直接计算或 Hessian-free 的方向导数估计，得到的 hvp 是相近的。对于较大规模的问题，当 Hessian 矩阵难以在短时间内求解时，采用方向导数来计算 hvp 是更优的选择。

3.4.2. 验证集精度变化趋势

训练集的损失函数能在一定程度上反映下降速度，但过低的损失函数可能导致过拟合的情况，因此我们需要其他指标来选择更优的参数。一般情况下，我们选择的标准是验证集上模型的预测精度。

图27展示了不同方法在验证集上的精度随迭代次

Table 1. Test Accuracy for Q3

Method	Test accuracy(%)
SGD	80
SGD-momentum	80
SGD-nesterov-momentum	80
Adagrad	85
RMSprop	80
Adam	85
AdaMax	85
NAdam	85
Newton	85
Hessian-free	85
BFGS	80
BFGS-momentum	70

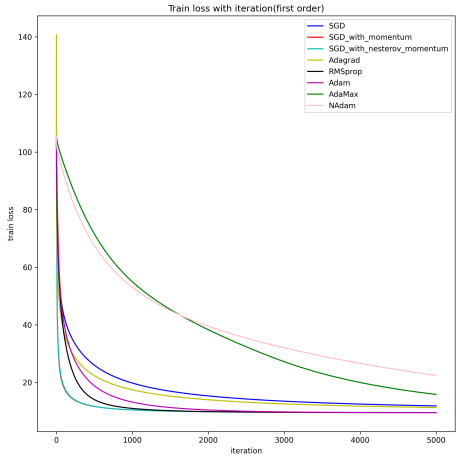

Fig. 25. 一阶方法的损失函数变化趋势

数的变化趋势。其中 Hessian-free、BFGS、AdaMax、NAdam 最终获得 96.7% 的精度，而其余方法都达到了 100%。总体来说，这些参数更新方法都是有效的，但在验证集上达到最优精度的时间和迭代次数有很大差异。从图上定性分析，大部分算法都在 500 次迭代之前收敛到比较好的参数，验证集精度能到 96% 以上，然后通过进一步参数更新，在 1000 次迭代左右达到 100% 的准确率。而一阶方法中的 AdaMax 和 NAdam、二阶方法中的 Hessian-free，在超过 1000 次迭代时，依然只有 60% 左右的精度，且 NAdam 算法直到 4000 多的迭代后，才收敛到其他算法在 1000 次内迭代的结果。深度学习中常用的 Adam 算法也在 2000 左右才达到最优精度，略逊于 SGD 的动量变体。定量方面，我们统计了每种方法在验证集上达到最优精度时迭代次数和时间开销，如表2所示。

定量结果在迭代次数方面与定性大致符合，但实际所花的时间差别较大。二阶方法的单次迭代普遍比一阶方法要慢很多：平均意义上，前者 0.006s 更新一次参数，而后者需要 0.036s，约为前者的 6 倍。这主要由于二阶方法寻找参数更新方向 d 时需要较复杂的运算：Newton(CG) 和 Hessian-free 都是共轭梯度法，每次都需要至多 N 步循环才能找到 d(N 表示 Hessian 方阵的维度)；而 BFGS 和其动量变体需要求解线性方程组和更新近似 Hessian 矩阵，时间复杂度也较高。相比于一阶方法每步更新只需计算梯度的低消耗，二阶方法每次都“大动干戈”只为求解一步 d，显然是不太合理的。在 3.4.4 中，我们将比较固定学习率的二阶方法和结合回溯线搜索的二阶方法之间的时间和精度。

3.4.3. 测试集精度

针对每种方法，我们选取验证集上的最优参数，并对测试集进行一步验证，得到表3。所有方法都得到比较理想的结果，96.7% 与 100% 仅仅相差一个实例的预测正误。这表明对于较小的数据集和不复杂的模型，一阶方法和二阶方法都可以在次数足够多的迭代后收敛到较为理想的结果。在实验中，我们也观察到 SGD 及其变体会更快寻找到较优参数，验证精度较高；而二阶方法普遍不太稳定，只有在迭代的后期才达到勉强接受的精度。如果训练时间有限或模型较为复杂，一阶方法中的 SGD 及其变体是可以信任的首选。

3.4.4. 二阶方法不同学习率 (步长) 设置的比较

在 3.4.2 中，我们提到针对二阶方法，采用较为复杂的方式来计算一步迭代的方向，在时间开销上是不合理的。传统 Newton(CG) 中寻找合适步长的方法为回溯线搜索，同时当牛顿减量满足一定条件时可以

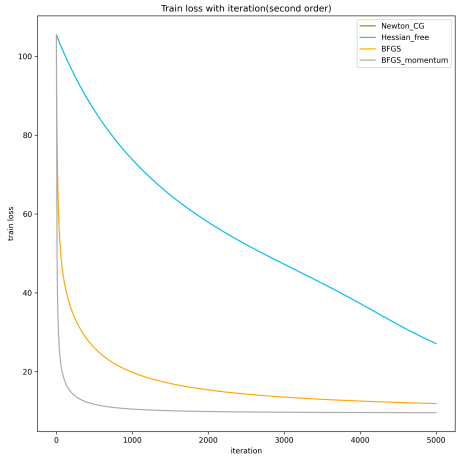

Fig. 26. 二阶方法的损失函数变化趋势

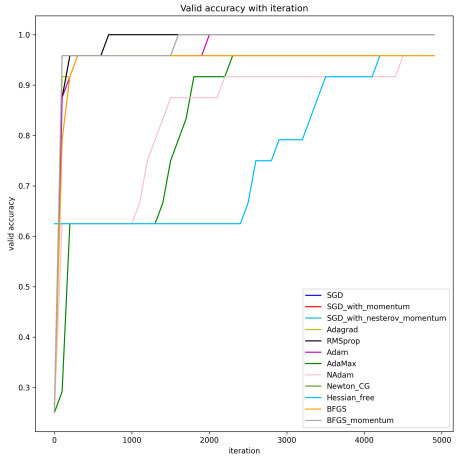

Fig. 27. 验证集精度变化趋势

提前结束整个迭代过程。我们将回溯线搜索融入提到的二阶方法中，并关注其在验证集上精度的变化趋势，如图28、29、30所示。其中 Newton(CG) 和 Hessian-free 重合，都在 6 次迭代左右达到最优精度，所花时间约为 0.20s，远小于 2.4.2 中报告的结果。动量加成的 BFGS 在 54 次左右第一次达到验证集上最优精度，但后期迭代中变化较为激烈，直到 200 次迭代后才保持稳定，所花时间约为 6.33s，也要优于之前的结果。而 BFGS 虽然前期以较少的迭代次数就实现验证集上 95% 左右的精度，但直到 2400 次迭代才达到 100% 的最优精度，所花时间为 121.75s，远高于之前报告的 11.31s。以上 4 种结合回溯线搜索的方法都在测试集上达到了 100% 的准确率，可见回溯线搜索能在保持精度的同时，加快二阶方法的收敛。针对数据规模不大的问题，如果能计算 Hessian 矩阵，使用结合 Line Search 的 Newton(CG) 或 Hessian-free 是不错的选择，其次可以选择动量加成的 BFGS，能确保收敛结果又快又好。

3.4.5. 一阶方法和二阶方法混合

一阶方法单次迭代速度较快，而二阶方法收敛效果比较好。将两种方法相结合可以扬长避短，最大化彼此的优势。实验中我们针对 NAdam 和 Newton(CG)，尝试了硬结合和软结合两种组合方式，主要考察在验证集上达到最优精度时迭代次数和所花时间。结果如表4所示。

注意到，这四种方法在测试集上精度相差不多，证明了其有效性，但迭代次数和时间开销大不相同。硬结合由于在进行一次迭代时需要同时计算一阶方向和二阶方向，所花时间自然比较长，而软结合随着迭代的进行，逐步从二阶方法转为一阶方法，且在训练后期不计算二阶方法，节省了大量时间。在迭代次数上，软结合也显著优于其他方法。

4. 总结

本文对深度学习中常用的优化器进行了全面的综述，详细介绍了 8 种一阶方法和 5 种二阶方法，并理论分析了 SGD 和 Adam 的收敛性。为了进一步综合一阶方法收敛速度快和二阶方法短时间收敛效果好的特点，采用硬软两种方式结合一、二阶方法。本文在 4 种数据集上进行了实验验证，总体来说，一阶方法要普遍优于二阶方法。得益于单步迭代的轻量性，一阶方法可以快速进行大量的迭代，并最终收敛到较优的局部极值。而二阶方法中有的需要计算 Hessian 矩阵，这对于深度学习中大规模参数矩阵而言是无法接受的；有的受超参影响较大，难以应用于未知结构的实际问题。如果问题规模较小且目标函数不复杂，可以考虑使用结合回溯线搜索的 Newton(CG) 或 Hessian-

Table 2. Best Iteration & Time

Method	Iteration	Time/s
SGD	300	1.87
SGD-momentum	1600	10.28
SGD-nesterov-momentum	1600	11.02
Adagrad	300	1.75
RMSprop	700	4.05
Adam	2000	12.33
AdaMax	2300	14.10
NAdam	4500	28.88
Newton(CG)	4200	102.49
Hessian-free	4200	174.33
BFGS	300	11.31
BFGS-momentum	1600	86.80

Table 3. Test Accuracy for Q4

Method	Test accuracy(%)
SGD	100
SGD-momentum	100
SGD-nesterov-momentum	100
Adagrad	100
RMSprop	100
Adam	96.7
AdaMax	96.7
NAdam	96.7
Newton(CG)	100
Hessian-free	96.7
BFGS	96.7
BFGS-momentum	100

free。在未来的工作中，我们将继续探索一阶方法在较为复杂的深度学习问题中的应用，并尝试结合最新的理论工具对优化器收敛性等问题作出分析。

Table 4. Best Iteration & Time & Test Accuracy

Method	Iteration	Time/s	Accuracy(%)
Newton	4200	97.15	100
NAdam	4500	24.74	96.7
hard	4400	102.88	100
soft	2200	12.11	100

5. 小组分工

龚畅：确立选题及项目思路并完成 16 周项目汇报；完成算法主体部分资料查找；完成 2.3 二分类问题及 2.4 多分类问题中的代码实现及相应论文编写；对论文整体进行检查和完善，编写摘要和总结；汇总代码。

来文博：完成算法理论分析部分资料查找；完成 1 算法主题及理论分析部分论文编写；完成 2.1 马鞍面逃离鞍点及 2.2 自造函数优化的代码实现及论文编写。

6. REFERENCES

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," 2015.

[2] ," .

[3] Thomas N. Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," 2016.

[4] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "Ten-

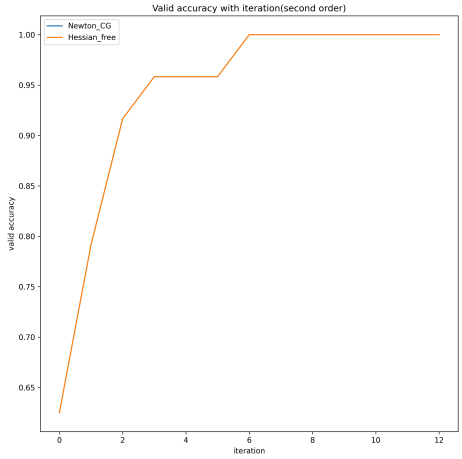

Fig. 28. Newton&Hessian(Line Search)

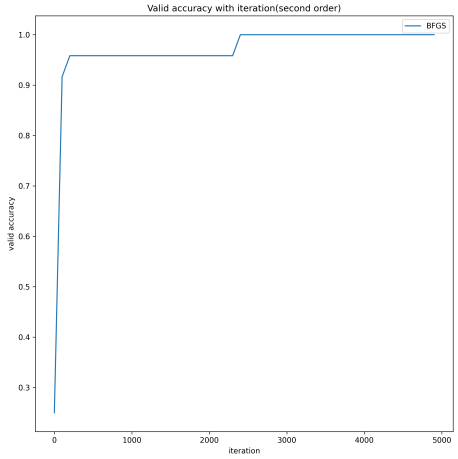

Fig. 29. BFGS(Line Search)

sorFlow: Large-scale machine learning on heterogeneous systems," 2015, Software available from tensorflow.org.

[5] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

[6] Sebastian Ruder, "An overview of gradient descent optimization algorithms," 2016.

[7] "深度学习算法收敛性证明之 sgd," https://zhuanlan.zhihu.com/p/338108328.

[8] Yurii Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence o(1/k2)," 1983.

[9] John Duchi, Elad Hazan, and Yoram Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, no. null, pp. 2121–2159, jul 2011.

[10] Geoffrey Hinton, "Overview of mini-batch gradient descent," http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[11] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," 2014.

[12] Timothy Dozat, "Incorporating Nesterov Momentum into Adam," in *Proceedings of the 4th International Conference on Learning Representations*, pp. 1–4.

[13] James Martens, "Deep learning via hessian-free optimization," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, Madison, WI, USA, 2010, ICML'10, p. 735–742, Omnipress.

[14] Duan Tao and Zhu, "Hessian-free second-order optimization on deep neural network," https://github.com/HessianFreeOptimization/HessianFreeOptimization/blob/master/10725_Final_hessianfreeoptimization.pdf.

[15] R. A. FISHER, "The use of multiple measure-

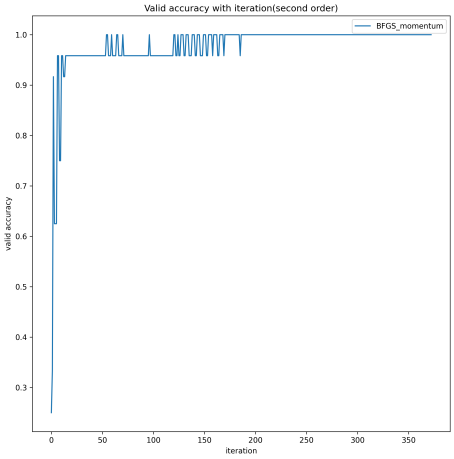

Fig. 30. BFGS-momentum(Line Search)

ments in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[16] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang, "JAX: composable transformations of Python+NumPy programs," 2018.