2025 春季优化讨论班

SGD 及演进

金璋

Apr, 2025

1. 背景
2. 随机梯度下降算法（SGD）
3. 动量加速原理与 Nesterov 算法
4. 自适应学习率机制
5. 随机梯度算法的收敛性分析

1 背景

2 随机梯度下降算法（SGD）

3 动量加速原理与 Nesterov 算法

4 自适应学习率机制

5 随机梯度算法的收敛性分析

监督学习模型

- 假定 (a, b) 服从概率分布 P ，其中 a 为输入，b 为标签.
- 我们的任务是要给定输入 a 预测标签 b ，即要决定一个最优的函数 ϕ 使得期望风险 $\mathbb{E}[L(\phi(a), b)]$ 最小，其中 $L(\cdot, \cdot)$ 表示损失函数，用来衡量预测的准确度，函数 ϕ 为某个函数空间中的预测函数.

实际问题中的近似

- 实际问题中我们不知道真实的概率分布 P ，而是随机采样得到一个数据集 $\mathcal{D} = \{(a_1, b_1), (a_2, b_2), \cdots, (a_N, b_N)\}$ 。数据集 $\mathcal{D}$ 对应经验分布

$$\hat{P} = \frac{1}{N}\sum_{n=1}^{N} \delta_{a_i, b_i}$$

其中 $\delta_{a,b}$ 表示 (a, b) 处的单点分布。

- 实际中为了缩小目标函数的范围，需要将 $\phi(\cdot)$ 参数化为 $\phi(\cdot; x) \cdot \phi$ 参数化的例子如线性函数、神经网络等.

随机优化问题

- 用经验风险来近似期望风险，即要求解下面的极小化问题：

$$\min_{x} \frac{1}{N}\sum_{i=1}^{N} L\left(\phi\left(a_i;x\right),b_i\right) = \mathbb{E}_{(a,b)\sim\hat{P}}[L(\phi(a;x),b)].$$

- 记

$$f_i(x) = L\left(\phi\left(a_i;x\right),b_i\right)$$

则只需考虑如下随机优化问题：

$$\min_{x\in\mathbb{R}^n} f(x) \stackrel{\text{def}}{=} \frac{1}{N}\sum_{i=1}^{N} f_i(x)$$

这也称为随机优化问题的有限和形式.

1 背景

2 随机梯度下降算法（SGD）

3 动量加速原理与 Nesterov 算法

4 自适应学习率机制

5 随机梯度算法的收敛性分析

梯度下降算法

- 为了讨论方便，我们先假设上式中的所有 $f_i(x)$ 是凸的、可微的．此时，可以运用梯度下降算法

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

来求解原始的优化问题．

$$\nabla f\left(x^k\right) = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i\left(x^k\right)$$

要计算这个梯度必须计算出所有的 $\nabla f_i\left(x^k\right)$．

- 然而在机器学习中，采集到的样本量是巨大的，因此计算 $\nabla f\left(x^k\right)$ 需要非常大的计算量．使用传统的梯度法求解机器学习问题并不是一个很好的做法．

随机梯度下降算法

- SGD 的基本迭代格式为

$$x^{k+1} = x^k - \alpha_k \nabla f_{s_k}\left(x^k\right)$$

  其中 s_k 是从 $\{1, 2, \cdots, N\}$ 中随机等可能地抽取的一个样本，α_k 称为步长. 在机器学习和深度学习领域中，更多的时候被称为学习率（learning rate）
- 随机梯度算法不去计算全梯度 $\nabla f\left(x^k\right)$，而是从众多样本中随机抽出一个样本 s_i，然后仅仅计算这个样本处的梯度 $\nabla f_{s_k}\left(x^k\right)$，以此作为 $\nabla f\left(x^k\right)$ 的近似.

小批量随机梯度法与随机次梯度法

- 实际计算中每次只抽取一个样本 s_k 的做法比较极端，常用的形式是小批量（mini-batch）随机梯度法。每次迭代中，随机选择一个元素个数很少的集合 $\mathcal{I}_k \subset \{1, 2, \cdots, N\}$，然后执行迭代格式

$$x^{k+1} = x^k - \frac{\alpha_k}{|\mathcal{I}_k|} \sum_{s \in \mathcal{I}_k} \nabla f_s \left(x^k\right)$$

- 当 $f_i(x)$ 是凸函数但不一定可微时，我们可以用 $f_i(x)$ 的次梯度代替梯度进行迭代。这就是随机次梯度算法。它的迭代格式为

$$x^{k+1} = x^k - \alpha_k g^k$$

其中 α_k 为步长，$g^k \in \partial f_{s_k}\left(x^k\right)$ 为随机次梯度，其期望为真实的次梯度.

梯度下降与微分方程

梯度下降的基本迭代格式为

$$x^{k+1} = x^k - \alpha \nabla f\left(x^k\right)$$

由于一般都有 $\alpha << 1$，因此可以改写为

$$\frac{x^{k+1} - x^k}{\alpha} = -\nabla f\left(x^k\right)$$

那么左边就近似于 x 的导数（假设它是时间 t 的函数），于是我们可以得到 ODE 动力系统：

$$\dot{\boldsymbol{x}} = -\nabla f(\boldsymbol{x})$$

因此，梯度下降算法实际上就是用欧拉解法去求解该动力系统，其最终可以收敛到一个不动点（令 $\dot{\boldsymbol{x}} = 0$），且稳定的不动点是一个极小值点。

随机梯度下降与随机微分方程

- SGD 的迭代公式为

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \gamma \nabla f_{\boldsymbol{R}}\left(\boldsymbol{x}^k\right)$$

  其中 R 代表全样本中的一个子集。注意 f 的最小值才是我们的目标，而 $f_{\boldsymbol{R}}$ 则是一个随机变量。$\nabla f_{\boldsymbol{R}}\left(\boldsymbol{x}^k\right)$ 只是 $\nabla f\left(\boldsymbol{x}^k\right)$ 的一个估计。

- 如果假设 $\nabla f_{\boldsymbol{R}}\left(\boldsymbol{x}^k\right) - \nabla f\left(\boldsymbol{x}^k\right) = \xi_n$ 服从一个方差为 σ^2 的正态分布（近似描述），那么随机梯度下降算法相当于在原动力系统重加入了高斯噪声：

$$\dot{\boldsymbol{x}} = -\nabla f(\boldsymbol{x}) + \sigma\xi$$

  其中 ξ 服从标准正态分布。该动力系统从 ODE 变成了 SDE。

随机梯度下降与随机微分方程

- 在噪声的高斯假设下，该方程的解的平衡状态的概率分布为

$$P(\boldsymbol{x}) \sim \exp\left(-\frac{f(x)}{\sigma^2}\right)$$

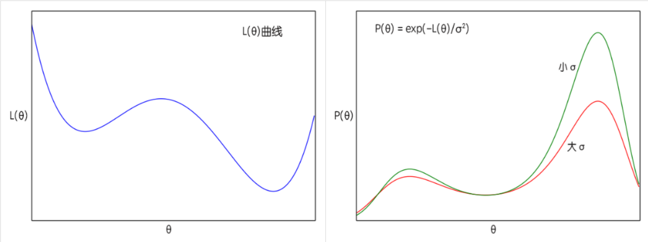
- 原来的 $f(x)$ 的极小值点变成了 $P(x)$ 的极大值点，如果无限长地执行随机梯度下降，理论上 x 能走遍所有可能的值，并且在 $f(x)$ 的各个“坑”中的概率更高。

DON'T DECAY THE LEARNING RATE, INCREASE THE BATCH SIZE

Samuel L. Smith*, Pieter-Jan Kindermans* & Quoc V. Le
Google Brain
{slsmith, pikinder, qvl}@google.com

随机梯度下降与随机微分方程

- σ^2 是梯度的方差，batch size 越大方差越小。σ^2 越大，说明 $P(\boldsymbol{x})$ 的图像越平缓，即越接近均匀分布，这时候 $\boldsymbol{x}$ 可能就到处跑；当 σ^2 越小时，原来 $f(\boldsymbol{x})$ 的极小值点的区域就越突出，这时候 $\boldsymbol{x}$ 就可能掉进某个坑里不出来了。



- 条件允许情况下，在使用 SGD 时，开始使用小 batch size 和大学习率，然后让 batch size 慢慢增加，学习率慢慢减小。

1 背景

2 随机梯度下降算法（SGD）

3 动量加速原理与 Nesterov 算法

4 自适应学习率机制

5 随机梯度算法的收敛性分析

动量方法

- 传统的梯度法在问题比较病态时收敛速度非常慢，随机梯度下降法也有类似的问题。为了克服这一缺陷，人们提出了动量方法（momentum），其思想是在算法迭代时一定程度上保留之前更新的方向，同时利用当前计算的梯度调整最终的更新方向.
- 动量方法的具体迭代格式如下：

$$
\begin{aligned}
v^{k+1} &= \mu_k v^k - \alpha_k \nabla f_{s_k}\left(x^k\right) \\
x^{k+1} &= x^k + v^{k+1}
\end{aligned}
$$

在计算当前点的随机梯度 $\nabla f_{s_i}\left(x^k\right)$ 后，我们并不是直接将其更新到变量 x^k 上，而是将其和上一步更新方向 v^k 做线性组合来得到新的更新方向 v^{k+1} .

动量方法

- 由动量方法迭代格式立即得出当 $\mu_k = 0$ 时该方法退化成随机梯度下降法. 在动量方法中，参数 μ_k 的范围是 $[0, 1)$ ，通常取 $\mu_k \geq 0.5$ ，其含义为迭代点带有较大惯性，每次迭代会在原始迭代方向的基础上做一个小的修正。
- 在普通的梯度法中，每一步迭代只用到了当前点的梯度估计，动量方法的更新方向还使用了之前的梯度信息。
- 当许多连续的梯度指向相同的方向时，步长就会很大，这从直观上看也是非常合理的。

动量方法的微分方程推导

我们考虑从原来的一阶微分方程推广到二阶，考虑一般的

$$\ddot{\boldsymbol{x}} + \lambda \dot{\boldsymbol{x}} = -\nabla f(\boldsymbol{x})$$

这样就真正对应一个（牛顿）力学系统了，其中 $\lambda > 0$ 引入了类似摩擦力的作用。从不动点的角度看，该方程最终收敛到的稳定不动点（$\ddot{\boldsymbol{x}} = \dot{\boldsymbol{x}} = 0$）也是 $f(x)$ 的一个极小值点.

动量方法的微分方程推导

将上式等价改写为

$$\dot{x} = \boldsymbol{\eta}, \quad \dot{\boldsymbol{\eta}} = -\lambda\boldsymbol{\eta} - \nabla f(\boldsymbol{x})$$

将 $\dot{x}$ 离散化，可以得到

$$\frac{\boldsymbol{x}_{n+1} - \boldsymbol{x}_n}{\gamma} = \boldsymbol{\eta}_{n+1/2}$$

$$\frac{\boldsymbol{\eta}_{n+1/2} - \boldsymbol{\eta}_{n-1/2}}{\gamma} = -\lambda\left(\frac{\boldsymbol{\eta}_{n+1/2} + \boldsymbol{\eta}_{n-1/2}}{2}\right) - \nabla f(\boldsymbol{x}_n)$$

动量方法的微分方程推导

记

$$\boldsymbol{v}_{n+1} = \gamma \boldsymbol{\eta}_{n+1/2}, \quad \beta = \frac{1-\lambda\gamma/2}{1+\lambda\gamma/2}, \quad \alpha = \frac{\gamma^2}{1+\lambda\gamma/2}$$

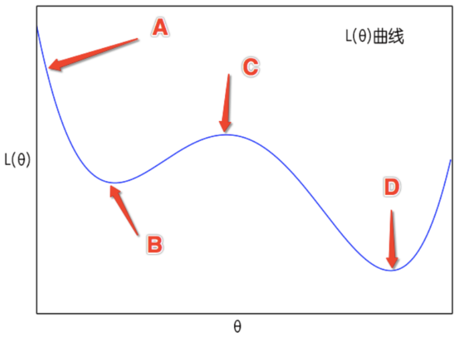
则有

$$\begin{aligned} \boldsymbol{x}_{n+1} &= \boldsymbol{x}_n + \boldsymbol{v}_{n+1} \\ \boldsymbol{v}_{n+1} &= \beta \boldsymbol{v}_n - \alpha \nabla f(\boldsymbol{x}_n) \end{aligned}$$

这就是带 Momentum 的 GD 算法的标准形式。

如何加速？

选定学习率 α 后，在同样精度下，Momentum 实际上是步长 $\sqrt{\alpha}$ 前进的，而纯 GD 则是以步长 α 前进的。由于学习率一般小于 1 ，所以 $\sqrt{\alpha} > \alpha$. 因此，Momentum 加速的原理就是可以在同等学习率、不损失精度的情况下，使得整个算法以更大步长前进.



从A点出发，sgd+momentum能够到达D点，而sgd通常只能到达B点

Nesterov 算法

假设 $f(x)$ 为光滑的凸函数．针对凸问题的 Nesterov 加速算法为

$$y^{k+1} = x^k + \mu_k \left(x^k - x^{k-1}\right)$$
$$x^{k+1} = y^{k+1} - \alpha_k \nabla f \left(y^{k+1}\right)$$

针对光滑问题的 Nesterov 加速算法迭代的随机版本为

$$y^{k+1} = x^k + \mu_k \left(x^k - x^{k-1}\right),$$
$$x^{k+1} = y^{k+1} - \alpha_k \nabla f_{s_k} \left(y^{k+1}\right),$$

其中 $\mu_k = \frac{k-1}{k+2}$，步长 α_k 是一个固定值或者由线搜索确定。

Nesterov 算法与动量方法的联系

若在第 k 步迭代引入速度变量 $v^k = x^k - x^{k-1}$ ，再合并原始 Nesterov 加速算法的两步迭代可以得到

$$x^{k+1} = x^k + \mu_k\left(x^k - x^{k-1}\right) - \alpha_k \nabla f_k\left(x^k + \mu_k\left(x^k - x^{k-1}\right)\right)$$

定义有关 v^{k+1} 的迭代式

$$v^{k+1} = \mu_k v^k - \alpha_k \nabla f_k\left(x^k + \mu_k v^k\right)$$

于是得到关于 x^k 和 v^k 的等价迭代：

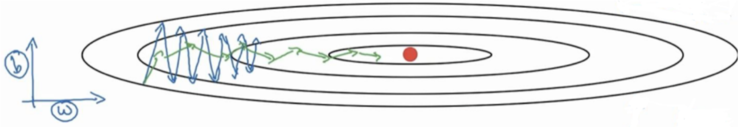
$$\begin{aligned} v^{k+1} &= \mu_k v^k - \alpha_k \nabla f_{s_k}\left(x^k + \mu_k v^k\right) \\ x^{k+1} &= x^k + v^{k+1} \end{aligned}$$

二者的主要差别在梯度的计算上．Nesterov 加速算法先对点施加速度的作用，再求梯度，可以理解为对标准动量方法做了校正．

1. 背景
2. 随机梯度下降算法（SGD）
3. 动量加速原理与 Nesterov 算法
4. 自适应学习率机制
5. 随机梯度算法的收敛性分析

AdaGrad

- 在一般的随机梯度法中，调参是一个很大的难点．我们希望算法能在运行的过程中，根据当前情况自发地调整参数。
- 对无约束光滑凸优化问题，点 x 是问题的解等价于该点处梯度为零向量．但梯度的每个分量收敛到零的速度是不同的。传统梯度算法只有一个统一的步长 α_k 来调节每一步迭代，它没有针对每一个分量考虑。
- 当梯度的某个分量较大时，可以推断出在该方向上函数变化比较剧烈，要用小步长；当梯度的某个分量较小时，在该方向上函数比较平缓，要用大步长．AdaGrad 就是根据这个思想设计的.



AdaGrad

令 $g^k = \nabla f_{s_k}(x^k)$，为了记录整个迭代过程中梯度各个分量的累积情况，引入向量

$$G^k = \sum_{i=1}^{k} g^i \odot g^i$$

从 G^k 的定义可知 G^k 的每个分量表示在迭代过程中，梯度在该分量处的累积平方和．当 G^k 的某分量较大时，我们认为该分量变化比较剧烈，因此应采用小步长，反之亦然．因此 AdaGrad 的迭代格式为

$$x^{k+1} = x^k - \frac{\alpha}{\sqrt{G^k + \varepsilon \mathbf{1}_n}} \odot g^k$$

$$G^{k+1} = G^k + g^{k+1} \odot g^{k+1}$$

RMSProp

RMSProp（Root Mean Square Propagation）是对 AdaGrad 的一个改进，该方法在非凸问题上可能表现更好．AdaGrad 会累加之前所有的梯度分量平方，这就导致步长是单调递减的，因此在训练后期步长会非常小，计算的开销也较大．RMSProp 提出只需使用离当前迭代点比较近的项，同时引入衰减参数 ρ ．具体地，令

$$M^{k+1} = \rho M^k + (1-\rho) g^{k+1} \odot g^{k+1}$$

再对其每个分量分别求根，就得到均方根（root mean square）

$$R^k = \sqrt{M^k + \varepsilon \mathbf{1}_n}$$

最后将均方根的倒数作为每个分量步长的修正．

RMSProp

RMSProp 迭代格式为：

$$
\begin{aligned}
x^{k+1} &= x^k - \frac{\alpha}{R^k} \odot g^k \\
M^{k+1} &= \rho M^k + (1-\rho) g^{k+1} \odot g^{k+1}
\end{aligned}
$$

引入参数 ε 同样是为了防止分母为 0 的情况发生．一般取 $\rho = 0.9$ ，$\alpha = 0.001$ 。可以看到 RMSProp 和 AdaGrad 的唯一区别是将 G^k 替换成了 M^k ．

AdaDelta

AdaDelta 在 RMSProp 的基础上，对历史的 Δx^k 也同样累积平方并求均方根：

$$
\begin{aligned}
D^k &= \rho D^{k-1} + (1-\rho)\Delta x^k \odot \Delta x^k \\
T^k &= \sqrt{D^k + \varepsilon \mathbf{1}_n}
\end{aligned}
$$

然后使用 T^{k-1} 和 R^k 的商对梯度进行校正：

$$
\begin{aligned}
\Delta x^k &= -\frac{T^{k-1}}{R^k} \odot g^k \\
x^{k+1} &= x^k + \Delta x^k
\end{aligned}
$$

AdaDelta

算法 8.13 AdaDelta

1. 输入 x^1, ρ, ε.
2. 置初值 $M^0 = 0, D^0 = 0$.
3. **for** $k = 1, 2, \cdots, K$ **do**
4. 随机选取 $i \in \{1, 2, \cdots, N\}$，计算梯度 $g^k = \nabla f_i(x^k)$.
5. 计算 $M^k = \rho M^{k-1} + (1-\rho) g^k \odot g^k$.
6. 计算 $\Delta x^k = -\dfrac{T^{k-1}}{R^k} \odot g^k$.
7. 计算 $D^k = \rho D^{k-1} + (1-\rho) \Delta x^k \odot \Delta x^k$.
8. $x^{k+1} \leftarrow x^k + \Delta x^k$.
9. **end for**

注意，计算步长时 T 和 R 的下标相差 1，这是因为我们还没有计算出 Δx^k 的值，无法使用 T^k 计算. AdaDelta 的特点是步长选择较为保守，同时也改善了 AdaGrad 步长单调下降的缺陷.

Adam

Adam（Adaptive Moment estimation）本质上是带动量项的 RMSProp。Adam 先选择一个动量项进行更新：

$$S^k = \rho_1 S^{k-1} + (1-\rho_1)\, g^k.$$

类似 RMSProp，Adam 也会记录梯度的二阶矩：

$$M^k = \rho_2 M^{k-1} + (1-\rho_2)\, g^k \odot g^k$$

与原始动量方法和 RMSProp 的区别是，由于 S^k 和 M^k 本身带有偏差，Adam 在更新前先对其进行修正：

$$\hat{S}^k = \frac{S^k}{1-\rho_1^k}, \quad \hat{M}^k = \frac{M^k}{1-\rho_2^k},$$

这里 ρ_1^k, ρ_2^k 分别表示 ρ_1, ρ_2 的 k 次方. Adam 最终使用修正后的一阶矩和二阶矩进行迭代点的更新.

$$x^{k+1} = x^k - \frac{\alpha}{\sqrt{\hat{M}^k + \varepsilon \mathbf{1}_n}} \odot \hat{S}^k.$$

1 背景

2 随机梯度下降算法（SGD）

3 动量加速原理与 Nesterov 算法

4 自适应学习率机制

5 随机梯度算法的收敛性分析

收敛性分析

- 随机梯度算法具有不确定性，这样的算法会有收敛性吗？概括来说，随机梯度下降法的收敛性依赖于步长的选取以及函数 f 本身的性质，在不同条件下会有不同结果.
- 我们先考虑一般凸函数下梯度算法的收敛性，施加如下假设：
  - 每个 $f_i(x)$ 是闭凸函数，存在次梯度
  - 随机次梯度二阶矩是一致有界的，即存在 M ，对任意的 $x \in \mathbb{R}^n$ 以及随机下标 s_k ，有

$$\mathbb{E}_{s_k}\left[\left\|g^k\right\|^2\right] \leq M^2 < +\infty, \quad g^k \in \partial f_{s_k}\left(x^k\right)$$

  - 迭代的随机点列 $\left\{x^k\right\}$ 处处有界，即 $\left\|x^k - x^*\right\| \leq R, \forall k$ ，其中 x^* 是问题的最优解.

引理

我们有如下重要的引理：

引理

在上述假设下，令 $\{\alpha_k\}$ 是任一正步长序列，$\{x^k\}$ 是由随机次梯度法产生的序列，那么对所有的 $K \geq 1$ ，有

$$\sum_{k=1}^{K} \alpha_k \mathbb{E}\left[f\left(x^k\right) - f\left(x^*\right)\right] \leq \frac{1}{2}\mathbb{E}\left[\left\|x^1 - x^*\right\|^2\right] + \frac{1}{2}\sum_{k=1}^{K} \alpha_k^2 M^2$$

引理的证明

令 $\bar{g}^k = \mathbb{E}\left[g^k \mid x^k\right], \xi^k = g^k - \bar{g}^k$. 由随机次梯度法的性质，

$$\bar{g}^k = \mathbb{E}\left[g^k \mid x^k\right] \in \partial f\left(x^k\right)$$

由次梯度的性质，

$$\left\langle \bar{g}^k, x^* - x^k \right\rangle \leq f\left(x^*\right) - f\left(x^k\right)$$

$$\begin{aligned}
&\left\|x^{k+1} - x^*\right\|^2 = \left\|x^k - \alpha_k g^k - x^*\right\|^2 \\
= &\left\|x^k - x^*\right\|^2 + 2\alpha_k \left\langle g^k, x^* - x^k \right\rangle + \alpha_k^2 \left\|g^k\right\|^2 \\
= &\left\|x^k - x^*\right\|^2 + 2\alpha_k \left\langle \bar{g}^k, x^* - x^k \right\rangle + \alpha_k^2 \left\|g^k\right\|^2 + 2\alpha_k \left\langle \xi^k, x^* - x^k \right\rangle \\
\leq &\left\|x^k - x^*\right\|^2 + 2\alpha_k \left(f\left(x^*\right) - f\left(x^k\right)\right) + \alpha_k^2 \left\|g^k\right\|^2 + 2\alpha_k \left\langle \xi^k, x^* - x^k \right\rangle
\end{aligned}$$

引理的证明

注意到 $\mathbb{E}\left[\xi^k \mid x^k\right]=0$ ，所以

$$\mathbb{E}\left[\left\langle \xi^k, x^*-x^k\right\rangle\right]=\mathbb{E}\left[\mathbb{E}\left[\left\langle \xi^k, x^*-x^k\right\rangle \mid x_k\right]\right]=0$$

对不等式两端求期望就得到

$$\alpha_k \mathbb{E}\left[f\left(x^k\right)-f\left(x^*\right)\right] \leq \frac{1}{2} \mathbb{E}\left[\left\|x^k-x^*\right\|^2\right]-\frac{1}{2} \mathbb{E}\left[\left\|x^{k+1}-x^*\right\|^2\right]+\frac{\alpha_k^2}{2} M^2$$

两边对 k 求和即得证.

随机次梯度算法的收敛性 1

根据该引理，我们很容易得到随机次梯度算法在收缩步长下的收敛性。

随机次梯度算法的收敛性 1（定理 1）

在收敛性假设的条件下，令 $A_K=\sum_{i=1}^{K}\alpha_i$，定义 $\bar{x}_K=\frac{1}{A_K}\sum_{k=1}^{K}\alpha_k x^k$，则

$$\mathbb{E}\left[f\left(\bar{x}_K\right)-f\left(x^*\right)\right]\leq\frac{R^2+\sum_{k=1}^{K}\alpha_k^2 M^2}{2\sum_{k=1}^{K}\alpha_k}$$

定理 1 的证明

由 $f(x)$ 的凸性以及引理得到

$$
\begin{aligned}
& A_k \mathbb{E}\left[f\left(\bar{x}_K\right)-f\left(x^*\right)\right] \\
\leq & \sum_{k=1}^K \alpha_k \mathbb{E}\left[f\left(x^k\right)-f\left(x^*\right)\right] \\
\leq & \frac{1}{2} \mathbb{E}\left[\left\|x^1-x^*\right\|^2\right]+\frac{1}{2} \sum_{k=1}^K \alpha_k^2 M^2 \\
= & \frac{R^2+\sum_{k=1}^K \alpha_k^2 M^2}{2}
\end{aligned}
$$

不等式两边同除以 A_K 得到

$$
\mathbb{E}\left[f\left(\bar{x}_K\right)-f\left(x^*\right)\right] \leq \frac{R^2+\sum_{k=1}^K \alpha_k^2 M^2}{2 A_K}
$$

随机次梯度算法的收敛性 1

从定理 1 可以看到，当

$$\sum_{k=1}^{\infty} \alpha_k = +\infty, \quad \frac{\sum_{k=1}^{K} \alpha_k^2}{\sum_{k=1}^{K} \alpha_k} \to 0$$

时，随机次梯度算法收敛。对一个固定的步长 α ，不等式右侧有一个不随 K 递减的常数，因此固定步长随机次梯度算法在函数值取期望意义下是不收敛的，它仅仅能找到一个次优解：

$$\mathbb{E}\left[f\left(\bar{x}_K\right) - f\left(x^*\right)\right] \leq \frac{R^2}{2K\alpha} + \frac{\alpha M^2}{2}$$

特别地，对于给定的迭代次数 K ，选取固定步长 $\alpha = \frac{R}{M\sqrt{K}}$ ，可以达到 $\mathcal{O}(1/\sqrt{K})$ 的精度，即

$$\mathbb{E}\left[f\left(\bar{x}_K\right) - f\left(x^*\right)\right] \leq \frac{RM}{\sqrt{K}}$$

随机次梯度算法的收敛性 2

在步长不增的情况下，我们可以得到直接平均意义下的收敛性.

随机次梯度算法的收敛性 2（定理 2）

在收敛性假设的条件下，令 $\{\alpha_k\}$ 是一个不增的正步长序列，$\bar{x}_K = \frac{1}{K}\sum_{k=1}^{K} x^k$，则

$$\mathbb{E}\left[f\left(\bar{x}_K\right) - f\left(x^*\right)\right] \leq \frac{R^2}{2K\alpha_K} + \frac{1}{2K}\sum_{k=1}^{K} \alpha_k M^2$$

随机次梯度算法的收敛性 2

注意该定理和定理 1 的不同之处在于 $\bar{x}_K$ 的定义．通过选取 $\mathcal{O}(1/\sqrt{k})$ 阶数的步长，我们可以得到目标函数的收敛速度为 $\mathcal{O}(1/\sqrt{k})$：

推论

在收敛性假设的条件下，令 $\alpha_k = \frac{R}{M\sqrt{k}}$，则

$$\mathbb{E}\left[f\left(\bar{x}_K\right) - f\left(x^*\right)\right] \leq \frac{3RM}{2\sqrt{K}}$$

其中 $\bar{x}_K$ 的定义和定理 2 相同．

我们可以发现随机次梯度算法和非随机次梯度算法具有相同的收敛速度——$\mathcal{O}(1/\sqrt{k})$，但随机次梯度算法每步的计算代价远小于非随机次梯度！

随机次梯度算法的收敛性 3

下面主要讨论随机次梯度算法在依概率意义下的收敛性和收敛速度：

定理 3

选择上述推论中的步长 α_k，使得 $\mathbb{E}\left[f\left(\bar{x}_K\right)-f\left(x^*\right)\right]\rightarrow 0$，那么我们有依概率收玫 $f\left(\bar{x}_K\right)-f\left(x^*\right)\xrightarrow{P}0(K\rightarrow\infty)$，即对任意的 $\varepsilon>0$，都有

$$\lim_{K\rightarrow\infty} P\left(f\left(\bar{x}_K\right)-f\left(x^*\right)\geq\varepsilon\right)=0$$

证明：由马尔可夫不等式立即得到

$$P\left(f\left(\bar{x}_K\right)-f\left(x^*\right)\geq\varepsilon\right)\leq\frac{1}{\varepsilon}\mathbb{E}\left[f\left(\bar{x}_K\right)-f\left(x^*\right)\right]\rightarrow 0.$$

随机次梯度算法的收敛性 4

定理 4

在假设收玫性假设的条件下，进一步假设对于所有的随机次梯度 g ，有 $\|g\| \leq M$. 那么对任意的 $\varepsilon > 0$ ，

$$f(\bar{x}_K) - f(x^*) \leq \frac{R^2}{2K\alpha_K} + \frac{1}{2K}\sum_{k=1}^{K}\alpha_k M^2 + \frac{RM}{\sqrt{K}}\varepsilon$$

以大于等于 $1 - e^{-\frac{1}{2}\varepsilon^2}$ 的概率成立，其中步长列 $\{\alpha_k\}$ 是单调不增序列，$\bar{x}_K$ 的定义和定理 2 中的定义相同.

可微强凸函数下随机梯度算法的收敛性

在前面的讨论中，我们知道对一般凸优化问题而言，随机（次）梯度下降法的收敛速度是 $\mathcal{O}\left(\frac{1}{\sqrt{K}}\right)$. 如果 $f(x)$ 有更好的性质，例如 $f(x)$ 是可微强凸函数，随机梯度下降法的收敛速度会有改善吗？首先我们列出假设：

- $f(x)$ 是可微函数，每个 $f_i(x)$ 梯度存在
- $f(x)$ 是梯度利普希茨连续的，相应常数为 L
- $f(x)$ 是强凸函数，强凸参数为 μ
- 随机梯度二阶矩是一致有界的，即存在 M，对任意的 $x \in \mathbb{R}^n$ 以及随机下标 s^k，有

$$\mathbb{E}_{s_k}\left[\|\nabla f_{s_k}(x)\|^2\right] \le M^2 < +\infty$$

随机梯度算法在固定步长下的收敛性分析

定理 6

在收敛性假设的条件下，定义 $\Delta_k = \left\| x^k - x^* \right\|$．对固定的步长 $\alpha_k = \alpha, 0 < \alpha < \frac{1}{2\mu}$，有

$$\mathbb{E}\left[f\left(x^{K+1}\right) - f\left(x^*\right)\right] \le \frac{L}{2}\mathbb{E}\left[\Delta_{K+1}^2\right] \le \frac{L}{2}\left[(1-2\alpha\mu)^K \Delta_1^2 + \frac{\alpha M^2}{2\mu}\right]$$

可以看到，对于固定的步长，算法不能保证收敛，这是因为的右端有不随 K 变化的常数.

随机梯度算法在递减步长下的收敛性分析

如果设置递减的步长，收敛阶可以达到 $\mathcal{O}(1/K)$.

定理 6

在上述定理的结果中，在收敛性假设的条件下，取递减的步长

$$\alpha_k = \frac{\beta}{k+\gamma}$$

其中 $\beta > \frac{1}{2\mu}, \gamma > 0$ ，使得 $\alpha_1 \leq \frac{1}{2\mu}$ ，那么对于任意的 $k \geq 1$ ，都有

$$\mathbb{E}\left[f\left(x^k\right) - f\left(x^*\right)\right] \leq \frac{L}{2}\mathbb{E}\left[\Delta_k^2\right] \leq \frac{L}{2}\frac{v}{\gamma+k}$$

这里

$$v = \max\left\{\frac{\beta^2 M^2}{2\beta\mu - 1}, (\gamma+1)\Delta_1^2\right\}$$