# 从梯度下降法到共轭梯度法

姚兴虎

2019 年 5 月 7 日

## 1 深度学习中的优化问题

深度学习算法在许多情况下都涉及优化，其中所涉及到的优化问题可描述为：寻找神经网络上的一组参数 $\boldsymbol{\theta}$, 他能显著地降低代价函数 $L(\boldsymbol{\theta})$, 该代价函数通常包括整个训练集上的性能评估和额外的正则化项。通常代价函数可写为在训练集上所有样本的平均误差，如：

$$L(\boldsymbol{\theta}) = \mathbb{E}_{(x,y)\sim\hat{p}_{\text{data}}} l(f(\mathbf{x};\boldsymbol{\theta}), y) \tag{1}$$

其中 $l$ 是每个样本的损失函数，$f(\mathbf{x};\boldsymbol{\theta})$ 是输入 $\mathbf{x}$ 时所预测的输出，$\hat{p}_{\text{data}}$ 是经验分布。监督学习中，$y$ 是目标输出。常见的每个样本的损失函数有平方损失，交叉熵损失等。理想的优化目标是在真实的数据分布 $p_{\text{data}}$ 上最小化损失函数。然而，我们遇到的机器学习问题通常是不知道数据的真实分布 $p_{\text{data}}(\mathbf{x}, y)$, 因此我们只能用训练集中的样本的经验分布 $\hat{p}_{\text{data}}(\mathbf{x}, y)$ 来进行优化求解。不考虑参数正则化等问题，神经网络的训练过程便可理解为一个目标函数为 $L(\boldsymbol{\theta})$ 的无约束优化问题，下面我们将介绍求解这一问题的几种经典优化方法。为了便于表述，本文所介绍的优化方法假定训练数据只有一个批量 (batch).

## 2 梯度下降法

### 2.1 基本梯度下降法

梯度下降法 (Gradient descent) 是当今神经网络训练中最常使用的一类优化算法。其基于以下的观察：如果实值函数 $F(\boldsymbol{\theta})$ 在 $\boldsymbol{\theta}_0$ 处可微且有定义，那么函数 $F(\boldsymbol{\theta})$ 在 $\boldsymbol{\theta}_0$ 点沿着梯度相反的方向 $-\nabla F(\boldsymbol{\theta}_0)$ 下降最快。我们首先在 $\boldsymbol{\theta}_0$ 上对 $F$ 进行泰勒展开，可以得到：

$$F(\boldsymbol{\theta}) \approx F(\boldsymbol{\theta}_0) + \nabla F(\boldsymbol{\theta}_0)^T(\boldsymbol{\theta} - \boldsymbol{\theta}_0). \tag{2}$$

我们希望寻找下一个迭代参数 $\boldsymbol{\theta}$ 使得 $F(\boldsymbol{\theta}) \leqslant F(\boldsymbol{\theta}_0)$, 由公式 (2) 可得，

$$F(\boldsymbol{\theta}) - F(\boldsymbol{\theta}_0) \approx \nabla F(\boldsymbol{\theta}_0)^T(\boldsymbol{\theta} - \boldsymbol{\theta}_0) \leqslant 0.$$

于是有:

$$\langle \nabla F(\boldsymbol{\theta}_0), \boldsymbol{\theta} - \boldsymbol{\theta}_0 \rangle \leqslant 0.$$

记参数的更新方向 $\mathbf{d} = \boldsymbol{\theta} - \boldsymbol{\theta}_0$, 则当 $\mathbf{d}$ 的方向与梯度的方向相反时，上诉内积达到最小。于是要想最小化目标函数，参数更新的方向便是当前梯度方向的反方向。

当找到了参数更新的方向时，我们还面临着在该方向上更新多大的步长的问题，步长选得太小会使得算法的收敛速度缓慢，步长选得太大会使得迭代过程震荡以至于不能收敛。在基本的梯度下降算法中，

我们通常经验地选取一个合适的步长或者让更新步长随着迭代次数的增加不断减小。当更新步长 $\alpha$ 确定后，我们便得到了基本的梯度下降迭代公式：

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \nabla F(\boldsymbol{\theta}_k) \tag{3}$$

其中 $\boldsymbol{\theta}_k$ 是当前的参数值，$\boldsymbol{\theta}_{k+1}$ 是更新后的参数值。

经典的梯度下降算法简单易懂，但是诸多缺点限制了其应用：(1) 优化步长需要经验地选取，这往往会给算法的实际应用造成麻烦，在应用时，我们需要精心寻找一个合适的步长来更新参数；(2) 当靠近极小值时，函数的梯度会变得特别小，这会使得在靠近极小值时参数的更新速度减慢；(3) 该方法只是对目标函数的局部线性近似进行优化，于是更新的方向可能不是当前目标函数的最优更新方向，可能会出现”之字形” 的更新过程。下节提出的最速下降法将对优化步长的选取问题提出一个解决方法。

### 2.2 最速下降法

基本梯度下降算法的一个缺陷在于更新步长 $\alpha$ 的选择，在一定的范围内沿着更新的负梯度的方向进行参数更新会使得函数值下降，但是当优化的步长太大时便有可能跳出函数值下降的范围。为此，我们可以将更新后的函数值 $F(\boldsymbol{\theta}_k - \alpha \nabla F(\boldsymbol{\theta}_k))$, 然后将其视为更新步长 $\alpha$ 的一元函数。通过在这个函数上利用一维搜索求解如下的优化问题得到更新步长 $\alpha_k$:

$$\alpha_k = \arg\min_{\alpha \geqslant 0} F(\boldsymbol{\theta}_k - \alpha \nabla F(\boldsymbol{\theta}_k)) \tag{4}$$

这样便能够在确定迭代方向的前提下，确定在该方向上使得目标函数值最小的迭代步长。针对梯度下降过程中的方向震荡问题，动量 (Momentum) 梯度下降方法通过对之前的所有累计的更新方向进行指数级的衰减移动平均，从而在一定程度上减少更新方向的不断震荡。

## 3 共轭梯度法

针对梯度下降法只是对函数的局部近似从而更新的方向可能并不是最优的方向这一缺陷，并且其收敛过程较慢，我们可以对目标函数进行二阶泰勒展开，然后利用牛顿法等二阶优化方法进行优化求解。牛顿法需要面临一个海森矩阵求逆的问题，一方面海森矩阵可能根本不可逆，另一方面大规模矩阵求逆需要庞大的计算量。多种拟牛顿法对海森矩阵进行近似在一定程度上解决了牛顿法所面临的问题。在这一节，我们介绍另一种一阶优化方法，其可以被认为是处于一阶梯度下降法和二阶牛顿法之间的算法，其计算复杂度要低于二阶方法但能减轻一些一阶优化算法中所面临的问题。

### 3.1 问题建立

共轭梯度法的目标是加速上文提到的最速下降法的收敛速度，同时避免牛顿法所带来的昂贵的计算代价。最初的共轭方向法是用来解决如下的无约束二次规划问题：

$$\min_{\mathbf{x} \in \mathbb{R}^n} \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{b}^T \mathbf{x} \tag{5}$$

其中 $\mathbf{x}$ 为待优化的向量，$\mathbf{A}$ 为对称正定矩阵，$\mathbf{b}$ 为已知向量。对公式进行求导并令导数等于零可得求解该问题等价于求解线性方程组

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \in \mathbb{R}^n \tag{6}$$

求解上述线性方程组的常规方法是通过高斯消元，但是这一方法计算复杂度比较高。共轭方向法可以通过最多 $n$ 步迭代来求解这一问题。

**定义 1** 记 $Q \in \mathbb{R}^{n\times n}$ 为一个对称矩阵，$d_1, d_2, \ldots, d_m \in \mathbb{R}^n$，若对于任意的 $i, j = 1, 2, \ldots, m$，有 $d_i^T Q d_j = 0, i \neq j$。则称 $d_1, d_2, \ldots, d_m$ 关于矩阵 $Q$ 相互共轭，$d_1, d_2, \ldots, d_m$ 称为 $Q$-共轭方向组。

在实际应用中，我们通常认为矩阵 $Q$ 是正定矩阵，但是在基本的定义中并没有对 $Q$ 进行这一约束。当 $Q = \mathbf{0}$ 时，任何两个向量都是共轭的；当 $Q$ 为单位阵时，共轭性等价于传统的向量正交性。下面的定理给出了共轭向量的一个性质。

**定理 1** 记 $Q$ 是一个对称正定矩阵，如果一组向量 $\{d_1, d_2, \ldots, d_m\}$ 是 $Q$-共轭的，那么这组向量是线性无关的。

我们可以记 $\mathbf{x} = \mathbf{x}^* + \mathbf{e}$, 其中 $\mathbf{x}^*$ 是优化问题的最优解，$\mathbf{e}$ 是我们想要减少的误差，于是我们可以通过减少下面的残量 $\mathbf{r}$ 这样误差项 $\mathbf{e}$ 可以通过我们的迭代过程不断缩小:

$$\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k = \mathbf{b} - \mathbf{A}\mathbf{x}^* - \mathbf{A}\mathbf{e}_k = -\mathbf{A}\mathbf{e}_k \tag{7}$$

## 3.2 共轭方向法

回顾优化目标 (5), 由于矩阵 $A$ 是对称正定的，因此可以充当一组向量的 $Q$-共轭矩阵。记 $\mathbf{x}^*$ 为优化问题的最优解，$\{\mathbf{d_0}, \mathbf{d_1}, \ldots, \mathbf{d_{n-1}}\}$ 是 $A$-共轭的，那么由定理 1可得:

$$\mathbf{x}^* - \mathbf{x}_0 = \alpha_0 \mathbf{d}_0 + \cdots + \alpha_{n-1}\mathbf{d}_{n-1} \tag{8}$$

其中 $\mathbf{x}_0$ 为迭代的初始点。从而我们有，$\mathbf{d}_i^T \mathbf{A}(\mathbf{x}^* - \mathbf{x_0}) = \mathbf{d}_i^T \mathbf{A}(\alpha_0 \mathbf{d}_0 + \cdots + \alpha_{n-1}\mathbf{d}_{n-1}) = \alpha_i \mathbf{d}_i^T \mathbf{A}\mathbf{d}_i$。因而我们便能够得到每次迭代的更新步长,

$$\begin{aligned}
\alpha_i &= \frac{\mathbf{d}_i^T \mathbf{A}(\mathbf{x}^* - \mathbf{x}_0)}{\mathbf{d}_i^T \mathbf{A}\mathbf{d}_i} \\
&= \frac{\mathbf{d}_i^T \mathbf{A}(\mathbf{x}^* - \mathbf{x}_i + \mathbf{x}_i - \mathbf{x}_0)}{\mathbf{d}_i^T \mathbf{A}\mathbf{d}_i} \\
&= \frac{\mathbf{d}_i^T \mathbf{A}(\mathbf{x}^* - \mathbf{x}_i)}{\mathbf{d}_i^T \mathbf{A}\mathbf{d}_i} + \frac{\mathbf{d}_i^T \mathbf{A}(\mathbf{x}_i - \mathbf{x}_0)}{\mathbf{d}_i^T \mathbf{A}\mathbf{d}_i} \\
&= \frac{\mathbf{d}_i^T \mathbf{A}(\mathbf{x}^* - \mathbf{x}_i)}{\mathbf{d}_i^T \mathbf{A}\mathbf{d}_i} + \frac{\mathbf{d}_i^T \mathbf{A}(\alpha_0 \mathbf{d}_0 + \cdots + \alpha_{n-1}\mathbf{d}_{i-1})}{\mathbf{d}_i^T \mathbf{A}\mathbf{d}_i} \\
&= \frac{\mathbf{d}_i^T \mathbf{A}(\mathbf{x}^* - \mathbf{x}_i)}{\mathbf{d}_i^T \mathbf{A}\mathbf{d}_i} + \mathbf{0} \\
&= -\frac{\mathbf{d}_i^T \mathbf{A}(\mathbf{x}_i - \mathbf{x}^*)}{\mathbf{d}_i^T \mathbf{A}\mathbf{d}_i} \\
&= -\frac{\mathbf{d}_i^T (\mathbf{A}\mathbf{x}_i - \mathbf{b})}{\mathbf{d}_i^T \mathbf{A}\mathbf{d}_i} \\
&= -\frac{\mathbf{d}_i^T \mathbf{g}_i}{\mathbf{d}_i^T \mathbf{A}\mathbf{d}_i}.
\end{aligned}$$

于是我们便可得到我们的迭代更新公式:

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \frac{\mathbf{d}_i^T \mathbf{g}_i}{\mathbf{d}_i^T \mathbf{A}\mathbf{d}_i}\mathbf{d}_i. \tag{9}$$

上述推导可以整理为如下的共轭方向定理:

**定理 2 (共轭方向定理)** 记向量 $\{\mathbf{d_0}, \mathbf{d_1}, \ldots, \mathbf{d_{n-1}}\}$ 是 $A$-共轭的，$\mathbf{x}_0 \in \mathbb{R}^n$ 是任意的一个 $n$ 维向量，则按照 $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{d}_i, \mathbf{g}_i = \mathbf{A}\mathbf{x}_i - \mathbf{b}, \alpha_i = -\frac{\mathbf{d}_i^T \mathbf{g}_i}{\mathbf{d}_i^T \mathbf{A}\mathbf{d}_i}$ 的迭代格式进行 $n$ 步迭代，我们就能得到 $\mathbf{x}_n = \mathbf{x}^*$。

按照上述迭代方法进行求解我们便可得到共轭方向法的迭代格式，值得注意的是，共轭方向法的共轭向量是通过矩阵 $\mathbf{A}$ 事先求出的。

### 3.3 共轭梯度法

上述共轭方向法需要事先提供一组 $\mathbf{Q}$ 共轭的向量，我们能否在计算的过程中不断生成一组共轭向量呢。本节介绍的共轭梯度法就可以基于当前和历史数据生成当前的共轭向量。

同样考虑如 (5) 所示的二次规划问题，我们选取最速下降法的更新方向来最为我们第一次迭代的更新方向，即：

$$\mathbf{d}_0 = -\mathbf{g}_0, \tag{10}$$

从而我们有:

$$\mathbf{x}_1 = \mathbf{x}_0 - \frac{\mathbf{g}_0^T\mathbf{d}_0}{\mathbf{d}_0^T\mathbf{A}\mathbf{d}_0}\mathbf{d}_0. \tag{11}$$

下一步我们利用 $\mathbf{g}_1$ 和 $\mathbf{d}_0$ 的线性组合来构造与 $\mathbf{d}_0$ 共轭的向量 $\mathbf{d}_1$, 即 $\mathbf{d}_1 = -\mathbf{g}_1 + \beta_0\mathbf{d}_0$, 一般来说，我们有：

$$\mathbf{d}_{i+1} = -\mathbf{g}_{i+1} + \beta_i\mathbf{d}_i \tag{12}$$

通过共轭约束求解可得:

$$\beta_i = \frac{\mathbf{g}_{i+1}^T\mathbf{A}\mathbf{d}_{i+1}}{\mathbf{d}_i^T\mathbf{A}\mathbf{d}_i} \tag{13}$$

这样我们便得到了更新方向的计算，将更新方向和更新步长带入迭代更新公式 (9), 我们便能得到共轭梯度法。

### 3.4 共轭梯度法的另一种推导

在第 $i$ 步迭代过程中，我们考虑含有 $i$ 个向量的矩阵 $V_i = [\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_i]$，我们记新的迭代目标点是当前点与所有之前迭代方向的线性组合的叠加，即 $\mathbf{x}_{i+1} = \mathbf{x}_i + V_i\mathbf{y}$, 其中 $\mathbf{y} = [y_1, y_2, \ldots, y_i]^T$ 是组合系数。我们可以写作 $V_i\mathbf{y} = \sum_{k=1}^{i} y_k\mathbf{d}_k$. 为了最小化 $f(\mathbf{x}_{i+1})$, 我们需要 $\nabla_y f(\mathbf{x}_{i+1}) = 0$，于是：

$$\begin{aligned}
\nabla_y f(\mathbf{x}_{i+1}) &= \nabla_y \left\{ \frac{1}{2}(\mathbf{x}_i + V_i\mathbf{y})^T A(\mathbf{x}_i + V_i\mathbf{y}) - \mathbf{b}^T(\mathbf{x}_i + V_i\mathbf{y}) \right\} \\
&= V_i^T A V_i\mathbf{y} + V_i^T A\mathbf{x}_i - V_i^T\mathbf{b} \\
&= V_i^T A V_i\mathbf{y} - V_k^T\mathbf{r}_k = 0
\end{aligned}$$

从而我们得到:

$$\mathbf{y} = (V_i^T A V_i)^{-1} V_i^T\mathbf{r}_k \tag{14}$$

进而我们可以得到更新后的新的函数值为:

$$\begin{aligned}
f(\mathbf{x}_{i+1}) &= f(\mathbf{x}_i) + \frac{1}{2}\mathbf{y}^T V_i^T A V_i\mathbf{y} + \mathbf{y}^T V_i^T(A\mathbf{x} - \mathbf{b}) \\
&= f(\mathbf{x}_k) - \frac{1}{2}\mathbf{r}_i^T V_i(V_i^T A V_i)^{-1} V_i^T\mathbf{r}_i.
\end{aligned}$$

更新后的残量为：

$$\begin{aligned}
\mathbf{r}_{i+1} &= \mathbf{b} - A\mathbf{x}_{i+1} = \mathbf{b} - A(\mathbf{x}_i + V_i(V_i^T A V_i)^{-1} V_i^T\mathbf{r}_i) \\
&= (I - A V_i(V_i^T A V_i)^{-1} V_i^T)\mathbf{r}_i.
\end{aligned}$$

此外，我们可以得到:

$$\begin{aligned} V_i^T \mathbf{r}_{i+1} &= V_i^T (I - AV_i (V_i^T AV_i)^{-1} V_i^T) \mathbf{r}_i \\ &= (V_i^T - V_i^T) \mathbf{r}_i = 0 \end{aligned}$$

即 $V_i^T \mathbf{r}_{i+1} = 0$。又由于 $V_{i-1}^T \mathbf{r}_i = 0$，于是我们还可以得到 $V_i^T \mathbf{r}_i = [0, 0, \ldots, 0, v_i^T \mathbf{r}_i]^T$ 这一方法的迭代过程如下所示:

1. 初始化 $i = 0, v_0 = \mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$.
2. Repeat:
3. $\quad \mathbf{x}_{i+1} = \mathbf{x}_i + V_i (V_i^T AV_i)^{-1} V_i^T \mathbf{r}_i, \qquad \mathbf{r}_{i+1} = \mathbf{b} - \mathbf{x}_{k+1}$;
4. $\quad$ 如果 $\|\mathbf{r}_{i+1}\| <$ tolerence, 退出循环;
5. $\quad$ 用 $V_k$ 和 $\mathbf{r}_{i+1}$ 计算 $v_{i+1}$
6. $\quad$ 将 $v_{i+1}$ 连接到 $V_i$ 上得到 $V_{i+1}$.
7. $\quad i = i + 1$.