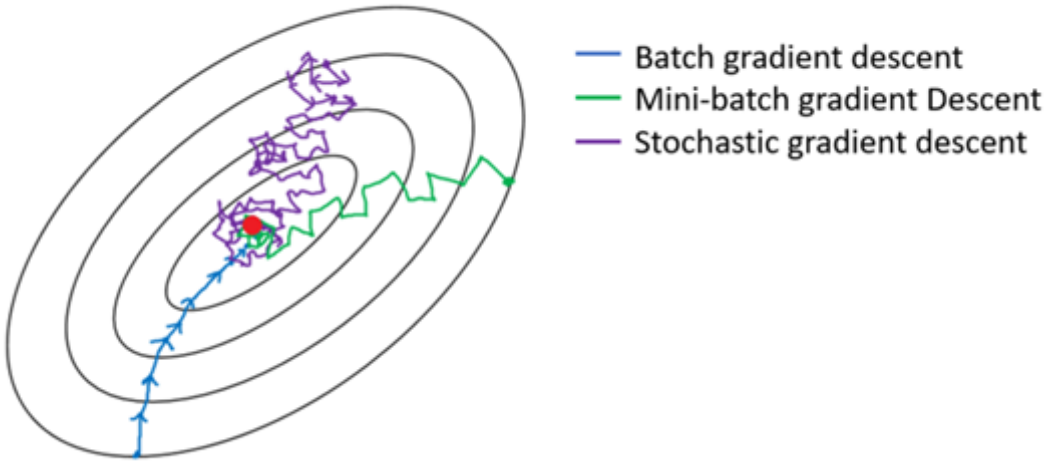
机器学习优化算法

作者：凯鲁嘎吉 - 博客园 http://www.cnblogs.com/kailugaji/



图来源：http://www.cs.virginia.edu/~hw5x/Course/RL2020-Fall/_site/static_files/ppt/policy-grad.pptx

1. 梯度下降法

在机器学习中，最简单、常用的优化算法就是梯度下降法，即通过迭代的方法来计算训练集$\mathcal{D}$上风险函数的最小值。

$$\theta_{t+1} = \theta_t - \alpha \frac{\partial \mathcal{R}_{\mathcal{D}}(\theta)}{\partial \theta} \tag{2.28}$$

$$= \theta_t - \alpha \cdot \frac{1}{N} \sum_{n=1}^{N} \frac{\partial \mathcal{L}\Big(y^{(n)}, f(\mathbf{x}^{(n)}; \theta)\Big)}{\partial \theta}, \tag{2.29}$$

其中θ_t为第t次迭代时的参数值，α为搜索步长。在机器学习中，α一般称为学习率（Learning Rate）。

梯度下降法（Gradient Descent Method），也叫最速下降法（Steepest Descend Method），经常用来求解无约束优化的极小值问题。

对于函数 $f(\mathbf{x})$，如果 $f(\mathbf{x})$ 在点 $\mathbf{x}_t$ 附近是连续可微的，那么 $f(\mathbf{x})$ 下降最快的方向是 $f(\mathbf{x})$ 在 $\mathbf{x}_t$ 点的梯度方法的反方向。

根据泰勒一阶展开公式，

$$f(\mathbf{x}_{t+1}) = f(\mathbf{x}_t + \triangle\mathbf{x}) \approx f(\mathbf{x}_t) + \triangle\mathbf{x}^{\mathrm{T}}\nabla f(\mathbf{x}_t). \tag{C.7}$$

要使得 $f(\mathbf{x}_{t+1}) < f(\mathbf{x}_t)$，就得使 $\triangle\mathbf{x}^{\mathrm{T}}\nabla f(\mathbf{x}_t) < 0$。我们取 $\triangle\mathbf{x} = -\alpha\nabla f(\mathbf{x}_t)$。如果 $\alpha > 0$ 为一个够小数值时，那么 $f(\mathbf{x}_{t+1}) < f(\mathbf{x}_t)$ 成立。

这样我们就可以从一个初始值 $\mathbf{x}_0$ 出发，通过迭代公式

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha_t\nabla f(\mathbf{x}_t),\ t \geq 0. \tag{C.8}$$

生成序列 $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$ 使得

$$f(\mathbf{x}_0) \geq f(\mathbf{x}_1) \geq f(\mathbf{x}_2) \geq \cdots \tag{C.9}$$

如果顺利的话，序列 $(\mathbf{x}_n)$ 收敛到局部最优解 $\mathbf{x}^*$。注意每次迭代步长 α 可以改变，但其取值必须合适，如果过大就不会收敛，如果过小则收敛速度太慢。

梯度下降法的过程如图C.1所示。曲线是等高线（水平集），即函数 f 为不同常数的集合构成的曲线。红色的箭头指向该点梯度的反方向（梯度方向与通过该点的等高线垂直）。沿着梯度下降方向，将最终到达函数 f 值的局部最优解。

梯度下降法为一阶收敛算法，当靠近极小值时梯度变小，收敛速度会变慢，并且可能以“之字形”的方式下降。如果目标函数为二阶连续可微，我们可以采用牛顿法。牛顿法为二阶收敛算法，收敛速度更快，但是每次迭代需要计算Hessian矩阵的逆矩阵，复杂度较高。

相反，如果我们要求解一个最大值问题，就需要向梯度正方向迭代进行搜索，逐渐接近函数的局部极大值点，这个过程则被称为梯度上升法（Gradient Ascent）。

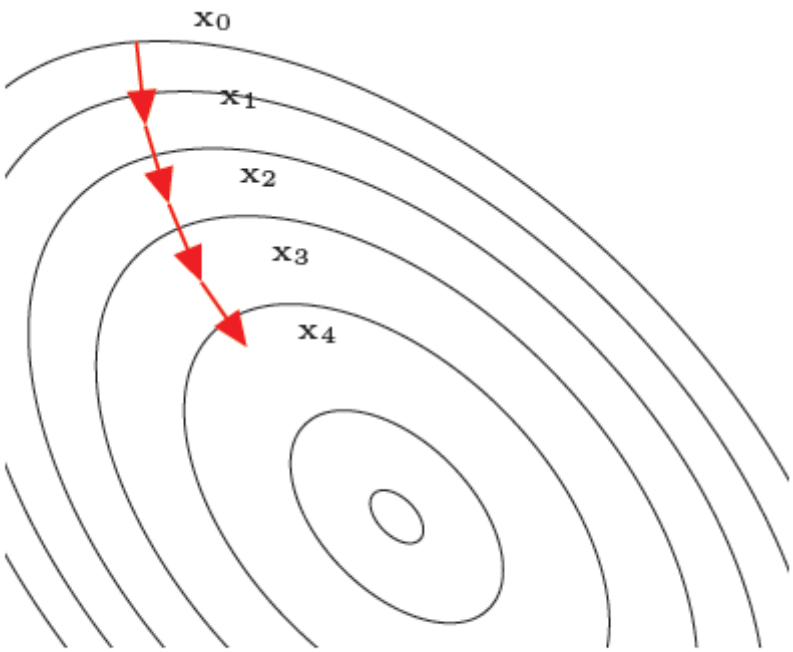


图 C.1 梯度下降法

2. 随机梯度下降法

在公式(2.28)的梯度下降法中，目标函数是整个训练集上风险函数，这种方式称为批量梯度下降法（Batch Gradient Descent，BGD）。批量梯度下降法在每次迭代时需要计算每个样本上损失函数的梯度并求和。当训练集中的样本数量N很大时，空间复杂度比较高，每次迭代的计算开销也很大。

在机器学习中，我们假设每个样本都是独立同分布的从真实数据分布中随机抽取出来的，真正的优化目标是期望风险最小。批量梯度下降相当于是从真实数据分布中采集N个样本，并由它们计算出来的经验风险的梯度来近似期望风险的梯度。为了减少每次迭代的计算复杂度，我们也可以在每次迭代时只采集一个样本，计算这个样本损失函数的梯度并更新参数，即随机梯度下降法（Stochastic Gradient Descent，SGD）。当经过足够次数的迭代时，随机梯度下降也可以收敛到局部最优解[Nemirovski et al., 2009]。

随机梯度下降法的训练过程如算法2.1所示。

批量梯度下降和随机梯度下降之间的区别在于每次迭代的优化目标是对所有样本的平均损失函数还是单个样本的损失函数。随机梯度下降因为实现简单，

收敛速度也非常快，因此使用非常广泛。随机梯度下降相当于在批量梯度下降的梯度上引入了随机噪声。当目标函数非凸时，反而可以使其逃离局部最优点。

算法 2.1: 随机梯度下降法

输入: 训练集 $\mathcal{D}=\{(\mathbf{x}^{(n)},y^{(n)})\}_{n=1}^{N}$，验证集 $\mathcal{V}$，学习率 α

随机初始化 θ;

repeat

  对训练集 $\mathcal{D}$ 中的样本随机重排序;

  for $n=1\cdots N$ **do**

    从训练集 $\mathcal{D}$ 中选取样本 $(\mathbf{x}^{(n)},y^{(n)})$;

    // 更新参数

    $\theta\leftarrow\theta-\alpha\dfrac{\partial\mathcal{L}\left(\theta;x^{(n)},y^{(n)}\right)}{\partial\theta}$;

  end

until 模型 $f(\mathbf{x};\theta)$ 在验证集 $\mathcal{V}$ 上的错误率不再下降;

输出: θ

3. 小批量梯度下降法

小批量梯度下降法 随机梯度下降法的一个缺点是无法充分利用计算机的并行计算能力。小批量梯度下降法（Mini-Batch Gradient Descent）是批量梯度下降和随机梯度下降的折中。每次迭代时，我们随机选取一小部分训练样本来计算梯度并更新参数，这样既可以兼顾随机梯度下降法的优点，也可以提高训练效率。

第 t 次迭代时，随机选取一个包含 K 个样本的子集 $\mathcal{I}_t$，计算这个子集上每个样本损失函数的梯度并进行平均，然后再进行参数更新。

$$\theta_{t+1} \leftarrow \theta_t - \alpha \cdot \frac{1}{K} \sum_{(\mathbf{x},y)\in\mathcal{I}_t} \frac{\partial \mathcal{L}\Big(y, f(\mathbf{x};\theta)\Big)}{\partial \theta}. \tag{2.30}$$

在实际应用中，小批量随机梯度下降方法有收敛快，计算开销小的优点，因此逐渐成为大规模的机器学习中的主要优化算法 [Bottou, 2010]。

4. 参考文献

[1] 邱锡鹏, 神经网络与深度学习[M]. 2019.

[2] [Machine Learning] 梯度下降法的三种形式BGD、SGD以及MBGD

[3] Ruder, Sebastian . "An overview of gradient descent optimization algorithms." (2016).