Deep Learning(IG, YB, AC) 学习笔记

李奥林

mrliaolin@outlook.com

目录

第一章 数值计算 **1**

1.1 上溢和下溢 1

1.2 病态条件数 1

1.3 基于梯度的优化方法 1

1.3.1 梯度之上：雅克比和海森矩阵 1

第二章 机器学习基础 **3**

2.1 随机梯度下降 3

第一章
数值计算

1.1 上溢和下溢

1.2 病态条件数

1.3 基于梯度的优化方法

在 $\boldsymbol{u}$（单位向量）方向的方向导数是函数 f 在 $\boldsymbol{u}$ 方向的斜率。方向导数是函数 $f(\boldsymbol{x}+\alpha\boldsymbol{u})$ 关于 α 的导数（在 $\alpha=0$ 时取得)。当 $\alpha=0$ 时，$\frac{\partial}{\partial\alpha}f(\boldsymbol{x}+\alpha\boldsymbol{u})=\boldsymbol{u}^\top\nabla_{\boldsymbol{x}}f(\boldsymbol{x})$。

1.3.1 梯度之上：雅克比和海森矩阵

二阶推导的前提是要最小化的函数能用二次函数很好地近似，此时海森的特征值决定了学习速率的量级。

可以通过方向二阶导数预期一个梯度下降步骤能表现得多好，在点 $\boldsymbol{x}^{(0)}$ 处作函数 $f(\boldsymbol{x})$ 的近似二阶泰勒级数：

$$f(\boldsymbol{x})\approx f(\boldsymbol{x}^{(0)})+(\boldsymbol{x}-\boldsymbol{x}^{(0)})^\top\boldsymbol{g}+\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}^{(0)})^\top\boldsymbol{H}(\boldsymbol{x}-\boldsymbol{x}^{(0)}) \tag{1.1}$$

如果我们使用学习速率 ϵ，那么新的点 $\boldsymbol{x}$ 将会是 $\boldsymbol{x}(0)-\epsilon\boldsymbol{g}$。代入上述的近似，可得

$$f(\boldsymbol{x}^{(0)}-\epsilon\boldsymbol{g})\approx f(\boldsymbol{x}^{(0)})-\epsilon\boldsymbol{g}^\top\boldsymbol{g}+\frac{1}{2}\epsilon^2\boldsymbol{g}^\top\boldsymbol{H}\boldsymbol{g} \tag{1.2}$$

函数在特定方向 $\boldsymbol{d}$ 上的二阶导数可以写成 $\boldsymbol{d}^\intercal\boldsymbol{H}\boldsymbol{d}$。当海森矩阵是正定的（所有特征值都是正的)，则对于任意方向，二阶导数均为正值，则临界点为局部最小点。当所有非零特征值是同号的且至少有一个特征值是 0 时，这个检测就是不确定的，这是因为单变量的二阶导数为 0 的点的二阶导数测试是不确定的。

其中 $\boldsymbol{g}^\top\boldsymbol{H}\boldsymbol{g}$ 可以表示 $\boldsymbol{H}$ 的特征值的式子（线性代数)。多维情况下海森的条件数(最大特征值和最小特征值的比值的绝对值）衡量这些二阶导数的变化范围。当海森的条

件数很差时，梯度下降法也会表现得很差。这是因为一个方向上的导数增加很快 (相对步长来说)，而在另一个方向上增加得很慢。梯度下降不知道导数的这种变化，所以它不知道应该优先探索导数长期为负的方向。病态条件数也导致很难选择合适的步长，步长必须足够小，以免冲过最小值而向具有较强的正曲率方向上升。这通常意味着步长太小，以致于在其它较小曲率的方向上进展不明显。

使用海森矩阵的信息来指导搜索可以解决这个问题，其中最简单的方法是牛顿法。当 f 是一个正定二次函数时，牛顿法只要应用一次就能直接跳到函数的最小点。如果 f 不是一个真正二次但能在局部近似为正定二次，牛顿法则需要迭代。迭代地更新近似函数和跳到近似函数的最小点可以比梯度下降更快地到达临界点。这在接近局部极小点时是一个特别有用的性质，但是在鞍点附近是有害的。当附近的临界点是最小点（海森的所有特征值都是正的）时牛顿法才适用，而梯度下降不会被吸引到鞍点（除非梯度指向鞍点)。

凸优化算法只对凸函数适用 -即海森处处半正定的函数。因为这些函数没有鞍点而且其所有局部极小点必然是全局最小点，所以表现很好。

第二章
机器学习基础

2.1 随机梯度下降

梯度下降的计算代价是 $O(m)$，随机梯度下降的核心是，梯度是期望。期望可使用小规模的样本近似估计。在算法的每一步，从训练集中均匀抽出一 minibatch 样本 $\mathbb{B} = \{\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(m')}\}$