Chapter 7: Optimization Methods

机器学习算法 = 模型表征 + 模型评估 + 优化算法

凸优化

- 凸函数曲面上任意两点连接而成的线段，其上任意一点都不会处于该函数曲面的下方
- 对于凸优化问题，所有的局部极小值都是全局极小值，因此这类问题一般认为是比较容易求解的问题
- 分类
  - SVM
  - 逻辑回归
  - 线性回归
- 非凸优化
  - PCA：可以借助SVD直接得到全局极小值
  - 低秩模型：如矩阵分解
  - DNN

经典优化算法

- 无约束优化问题
  - 直接法
    - 目标函数需要满足
      - $\nabla L(\theta^*) = 0$
      - Loss(θ)函数是凸函数，那么θ*是最优解的充分条件为Loss在θ*出的梯度为0
      - 上使有闭式解
    - 限制了引用范围
    - $L(\theta) = \|X\theta - y\|_2^2 + \lambda \|\theta\|_2^2$
    - 岭回归Ridge Regression
  - 迭代法
    - $\delta_t = arg\min_{\delta} L(\theta_t + \delta)$
    - 如果当前对最优解的估计值为θ_t，希望求解优化问题
    - 迭代地修正对最优解的估计
    - $\theta_{t+1} = \theta_t - \alpha \nabla L(\theta_t)$
    - 先用一阶泰勒展开目标函数，并对其求导为0时的值就是改进值
    - 梯度下降法，梯度就是目标函数的一阶信息
    - 一阶法
    - $\theta_{t+1} = \theta_t - \nabla^2 L(\theta_t)^{-1} \nabla L(\theta_t)$
    - 先用二阶泰勒展开目标函数，并对其求导为0时的值就是改进值
    - 牛顿法，Hessian矩阵就是目标函数的二阶信息
    - 通常收敛速度比一阶法快，但是高维情况下，Hessian矩阵求逆计算复杂度很大，而且当目标函数非凸时，二阶法有可能收敛到鞍点
    - 二阶法
    - 1970年由Charles George Broyden, Roger Fletcher, Donald Goldfarb和David Shanno提出的BFGS算法
    - 针对二阶法矩阵求逆的计算复杂度过高的问题
    - 1989年扩展为低存储的L-BFGS算法

L1正则化与稀疏性

- 稀疏性
  - 模型很多参数为0，相当于模型进行了一次特征选择，只留下一些比较重要的特征，提高模型的泛化能力，降低过拟合的可能
- 函数叠加
  - 通过对加入正则化项的目标函数求导，判断在原点出，即w=0的时候的导数是否大于等于0
  - L1正则化在原点左侧，如果原目标函数L(w)+C|w|的导数绝对值小于C，C是正则项的系数，则左侧导数是小于0，单调递减，右侧是大于0，单调递增，因此原点处是极小值，代表w=0是极小值的一个解，所以L1正则项会产生稀疏解，既有稀疏性
- 解空间形状
  - L2正则化
    - 因为必须保证L2函数不能大于m
    - 只有减小w绝对值得作用，对解空间的稀疏性没有贡献
  - L1正则化
    - L1正则项约束的解空间是多边形
    - 在线梯度算法中，往往采用阶段梯度下降法来产生稀疏性
  - 通过KKT条件可得,带正则项和带约束条件是等价的
    - 加入一个正则项就定义了一个解空间约束
    - 如果原问题目标函数的最优解不是恰好落在解空间类内，那么约束条件下的最优解一定是在解空间的边界上，而L1要角分明的解空间显然更容易与目标函数等高线在角点碰撞，从而产生稀疏解
- 贝叶斯先验
  - 最通俗易懂的解释
  - L1正则化相当于对模型w引入了拉普拉斯先验，L2正则化相当于引入了高斯先验，而拉普拉斯先验使得参数为0的可能性更大
  - L2假设w是高斯分布，高斯分布在极值点0点处是平滑的，代表着w在极值点处取不同值得可能性是接近的，即说明L2正则化只会让w更接近0，但是不等于0的可能性也大
  - L1假设w是拉普拉斯分布，拉普拉斯分布在极值点0点处是尖峰状，代表着w在极值点处取0的可能性非常高，即说明L1正则项会的使得w取得0的可能性更高

2018.12.18 Tuesday. By KuKuXia@github.com

ML/DL/RL深度交流讨论群: 622545311，加群时请备注: 姓名-方向-公司or高校

监督学习

- 损失函数刻画了模型和训练样本的匹配程度
- 损失函数分类
  - 分类问题，Y={1,1}
    - $L_{0-1}(f,y) = 1_{fy \le 0}$
    - 0-1损失
    - 非凸、非光滑，很难直接对其优化
    - $L_{hinge}(f,y) = max(0, 1-fy)$
    - Hinge损失函数
    - 是0-1损失函数相对紧的凸上界，且当fy>=1时，该函数不对其做任何处罚
    - fy=1时不可导，因此不能使用梯度下降法进行优化
    - 次梯度下降法(Subgradient Descent Method)
    - $L_{logistic}(f,y) = log_2(1 + exp(-fy))$
    - Logistic损失函数
    - 是0-1损失函数的凸上界，处处光滑，可以使用梯度下降法优化
    - 对所有样本点都有处罚，因此对异常值相对更敏感一些
    - $L_{cross\_entropy}(f,y) = -log_2((1+fy)/2)$
    - 交叉熵
    - 预测值取值为[-1,1]
  - 回归问题Y=R
    - $L_{square}(f,y) = (f-y)^2$
    - 平方损失函数
    - 对异常值敏感
    - 相当于在做均值回归
    - 光滑函数，可以使用SGD优化
    - $L_{abaolute}(f,y) = |f-y|$
    - 绝对损失函数
    - 当f=y处无法求导数
    - 相当与在做中值回归
    - 对异常点更鲁棒一些
    - $L_{Huber}(f,y) = \begin{cases} (f-y)^2, & |f-y| \le \delta \\ 2\delta|f-y| - \delta^2, & |f-y| > \delta \end{cases}$
    - Huber损失函数
    - 在|f-y|较小时为平方损失
    - 在|f-y|较大时为线性损失
    - 处处可导且对异常值鲁棒

优化问题的目标函数

- $L(\theta) = E_{(x,y)\sim P_{data}} L(f(x,\theta),y)$
- 优化问题的目标函数通常为：
- $\theta^* = arg\min_{\theta} L(\theta)$
- 求解的优化问题为找到平均损失最小的模型参数
- $L(\theta) = \frac{1}{N}\sum_{i=1}^{N} L(f(x_i,\theta),y_i)$
- 近似目标函数
- $\nabla L(\theta) = \frac{1}{M}\sum_{i=1}^{M} \nabla L(f(x_i,\theta),y_i)$
- 目标梯度函数

随机梯度下降法SGD： Stochastic Gradient Descent

- 经典的批量梯度下降法 BGD: (Batch Gradient Descent)
  - 梯度下降法需要读取所有的训练数据的平均损失来近似目标函数，对于大规模数据问题不适用
  - 正常下山，一次读取全部数据，看到全景，稳定地逼近最优点
- 随机梯度下降法 SGD： Stochastic Gradient Descent
  - 用随机抽取的单个训练样本的损失来近似平均损失，单个训练数据就可以对模型参数进行一次更新，大大加快了收敛速率，也适用于数据来源不断的在线更新场景
  - $L(\theta; x_i, y_i) = L(f(x_i,\theta), y_i)$
  - 近似目标函数
  - $\nabla L(\theta; x_i, y_i) = \nabla L(f(x_i,\theta), y_i)$
  - 目标梯度函数
  - 缺点
    - 不稳局部最优点，但山谷震荡和鞍点停滞
    - 蒙着眼睛下山，一次一个数据，对梯度的估计常常出现偏差，造成目标函数曲线收敛的很不稳定，伴有剧烈波动，有时甚至不收敛的情况
- 小批量梯度下降法：Mini-Batch Gradient Descent
  - $L(\theta) = \frac{1}{m}\sum_{j=1}^{m} L(f(x_j,\theta),y_j)$
  - 近似目标函数
  - $\nabla L(\theta) = \frac{1}{m}\sum_{j=1}^{m} \nabla L(f(x_j,\theta),y_j)$
  - 目标梯度函数
  - 为了降低单个样本随机梯度的方差，从而使得迭代算法更加稳定，也为了充分利用高度优化的矩阵运算操作，在实际中会同时处理m（m << M）个样本训练数据
  - 超参数
    - 参数m的选取
    - 超参数之一，根据不同问题而定，通常m取2的幂次能充分利用矩阵运算操作
    - 32,64,158,256等
    - 如何选择m个数据？
    - 遍历训练数据之前，先对所有数据进行随机排序，然后在每次迭代时按顺序挑选m个训练数据直至遍历完所有的数据
    - 学习速率的选择
    - 衰减学习率的方案，先大后小

随机梯度下降法的改进

惯性保持与环境感知

- 惯性保持：利用动量的方法
  - 模型参数的迭代公式为
  - $v_t = \gamma v_{t-1} + \eta g_t$
  - 前进步伐
  - $\theta_{t+1} = \theta_t - v_t$
  - 参数更新
  - 收敛速度更快，收敛曲线也更加稳定
- 环境感知
  - 是指在参数空间中，根据不同的参数的一些经验性判断，自适应地驱动参数的学习速率，不同参数的更新步幅是不同的
  - 在实际应用中，我们希望更新频率低的参数可以拥有较大的更新步幅，而更新频率高的参数的步幅可以减小
  - AdaGrad
  - 采用历史梯度平方和来衡量不同参数的梯度的稀疏性，取值越小表明越稀疏
- Adam
  - 结合了惯性保持与环境感知的优点
  - 记录梯度的一阶矩(first moment)
  - 过往梯度与当前梯度的平均，体现了惯性保持
  - 记录梯度的二阶矩(second moment)
  - 过往梯度平方与当前梯度平方的平均，类似AdaGrad方法，体现了环境感知能力，为不同参数产生自适应的学习速率
  - 一阶矩和二阶矩采用类似于滑动窗口内求平均的思想进行融合，即当前梯度和近一段时间内梯度的平均值，时间久的梯度对当前平均值的贡献呈指数衰减
  - $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$
  - m_t一阶矩
  - 一阶矩和二阶矩采用指数衰退平均的技术，β_1与β_2是衰减系数，计算公式为：
  - $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
  - v_t二阶矩
  - 更新公式
  - $\theta_{t+1} = \theta_t - \frac{\eta * M_t}{\sqrt{V_t} + \epsilon}$
  - $M_t = \frac{m_t}{1-\beta_1^t}, V_t = \frac{V_t}{1-\beta_2^t}$
- 其他方法
  - AdaDelta
  - RMSProp
  - AdaMax
  - Nadam
  - Nesterov Accelerated Gradient