最优化算法

杨周旺

中国科学技术大学
数学科学学院

2021年3月

Outline I

Description

- The course is devoted to the mathematical fundamentals of optimization and the practical algorithms of optimization.
- The course covers the topics of nonlinear continuous optimization, sparse optimization, and optimization methods for machine learning.

Objectives

Objectives of the course are

- to develop an understanding of the fundamentals of optimization;
- to learn how to analyze the widely used algorithms for optimization;
- to become familiar with the implementation of optimization algorithms.

Prerequisites

- Knowledge of Linear Algebra, Real Analysis, and Mathematics of Operations Research are very important for this course.
- Simultaneously, the ability to write computer programs of algorithms is also required.

Topics Covered

- Unconstrained Optimization
- Constrained Optimization
- Convex Optimization
- Sparse Optimization
- Optimization Methods for Large-scale Machine Learning

Textbook and References

1 R. Fletcher. Practical Methods of Optimization (2nd Edition), John Wiley & Sons, 1987.

2 J. Nocedal and S. J. Wright. Numerical Optimization (2nd Edition), Springer, 2006.

3 S. Boyd and L. Vandenberghe. Convex Optimization, Cambridge University Press, 2004.

4 M. Elad. Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing. Springer, 2010.

5 L. Bottou, F.E. Curtis, J. Nocedal. Optimization methods for large-scale machine learning. SIAM Review, 60(2): 223-311, 2018.

Grading

(1) Homework (10%)

(2) Project (40%)

(3) Final Exam (50%)

Project

- 如下题目三选二，编程语言不限。
  - 实现一种求解大规模线性方程的求解器，其系数矩阵为稀疏矩阵。
  - 实现一种线路设计自适应优化算法，优化内容包括线路设计合理性、材料及施工成本等。
  - 针对大规模机器学习模型，实现一种随机梯度类算法。
- 要求提交程序代码，用户指南及对应的测试报告。

提交要求

- 除程序本体外，需要完成一份作业文档，内容包括但不限于算法原理、数据集说明、程序输入输出说明、程序测试、分析总结。
- 在课程网站上提交一个包含程序代码和作业文档的zip压缩包文件

线性方程组

线性方程组

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_1 \end{cases}$$

简记为$Ax = b$，其中

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

求解线性方程组有两大类方法：直接法和迭代法。

线性方程组：直接法

- 若A为对角阵，求解很简单，$x_i = b_i/a_{ii}$。
- 若A为下三角阵

$$A = \begin{bmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ \vdots & \vdots & \ddots & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$

则可以从第1个方程解得x_1，将x_1代入第2个方程解得x_2，将x_1, x_2代入第3个方程解得x_3，…，将$x_1, x_2, \dots, x_{i-1}$代入第i个方程解得x_i，依次求得$x_1, \dots, x_n$。

- 若A为上三角阵，与下三角阵的情形相似，但从x_n解起。

LU分解

- 对一般的A，假如A可以分解为一个下三角阵L和一个上三角阵U之积：$A = LU$，则求解方程组$Ax = b$可分成两步：$Lz = b$解得z，$Ux = z$解得x。并不是所有方阵都能执行LU分解。LU分解本质上是**高斯消元法**。
- 当A为实对称正定阵时，可以取得$U^T = L$，此时称为Cholesky分解。
- 为保持系数矩阵的稀疏性，对一般稀疏矩阵的LU分解需要仔细考察消元顺序。

线性方程组：迭代法

指定某个称为**分裂矩阵**的矩阵Q，并把原问题改写成等价形式：

$$Qx = (Q - A)x + b,$$

由此得到一个迭代过程：

$$Qx^{(k)} = (Q - A)x^{(k-1)} + b \qquad (k \geq 1).$$

假设Q可逆，则方程组的解将满足

$$x = (I - Q^{-1}A)x + Q^{-1}b$$

因此

$$(x^{(k)} - x) = (I - Q^{-1}A)(x^{(k-1)} - x)$$

根据压缩映射原理，$\|I - Q^{-1}A\| < 1$时，对任意初值$x^{(0)}$该迭代过程收敛。（**注意：收敛性不能保证，因系数矩阵而异。**）

一些分裂矩阵的取法

- 理查森方法：取$Q = I$
- 雅可比方法：取Q为对角阵，其对角元与A的对角元相同
- 高斯-赛德尔方法：取Q为A的下三角部分包括对角线

外推技巧 外推是在迭代过程中引进参数$\gamma \neq 0$，并将迭代过程改写为

$$Qx^{(k)} = (Q - A)x^{(k-1)} + b + (1 - \gamma)Qx^{(k-1)}$$

若关于G的特征值仅知道位于区间$[a, b]$中且$1 \notin [a, b]$，则γ的最好选择是$2/(2 - a - b)$。这里$G \triangleq I - Q^{-1}A$。

线性方程组：基于优化的迭代法

当A是一个$n \times n$对称正定阵时，求解$Ax = b$等价于求解问题$\min_x \frac{1}{2}x^T Ax - b^T x$，因此可以使用优化算法求解，例如最速下降法、共轭梯度法。

- 最速下降法的表现显著差于共轭梯度法，基本只具有理论上讨论的价值；
- 共轭梯度法及其变种是被广泛使用的求解方法，著名的线性代数eigen库便包含求解线性方程的共轭梯度法。

以上优化方法的具体过程参照课程讲义。

稀疏矩阵存储

对于稀疏矩阵，只需要保存非零元素的位置和值。对于矩阵A，记其非零元个数为$nnz(A)$，行数为m。

- **COO** Coordinate格式，使用三个数组value, row, col保存稀疏矩阵，其分量分别为第i个非零元素的取值和所在的行、列。需要$3nnz(A)$的存储空间。
- **CSR** Compressed Sparse Row格式，使用三个数组value, col, rowIndex保存稀疏矩阵，其中前两个数组的各分量分别为第i个非零元素的取值和所在的列，而rowInd存储每一行的第一个非零元素在value中的序号。需要$2nnz(A)+m$的存储空间。

稀疏矩阵存储

例：对矩阵

$$A = \begin{bmatrix} 1 & 0 & 3 & 0 \\ 0 & 0 & 6 & 0 \\ 3 & 0 & 0 & 2 \\ 0 & 0 & 0 & 5 \end{bmatrix}$$

使用COO格式存储为

序号	0	1	2	3	4	5
value	1	3	6	3	2	5
row	0	0	1	2	2	3
col	0	2	2	0	3	3

稀疏矩阵存储

使用CSR格式存储为（注意对比COO格式中的row）

序号	0	1	2	3	4	5
value	1	3	6	3	2	5
col	0	2	2	0	3	3

序号	0	1	2	3
rowInd	0	2	3	5

稀疏矩阵存储

其他存储格式：

- CSC, Compressed Sparse Column格式，与CSR类似，但是行和列的角色互换
- BSR, Block Compressed Sparse Row格式，以类似CSR的方式存储分块稀疏矩阵
- ELLPACK, 通过两个行数与原矩阵A相同的矩阵col和$value$存储，col和$value$中的每一行分别为原矩阵A中该行非零元素对应的列和取值。非零元素较少的行，行末用占位符如“$*$”补齐。
- DIA, Diagonal格式，使用行数等于原矩阵A行数的矩阵$diagonals$存储，列代表原矩阵的对角线，行代表原矩阵的行，从原矩阵的左下到原矩阵的右上依次存储，省略全零对角线，不从第一行开始或不结束于最后一行的对角线，使用占位符补齐。

作业要求

- 程序本身，或程序中的函数，至少需要以某种稀疏存储格式存储的稀疏矩阵和线性方程右端项为输入，即以$Ax = b$中的A和b为输入。允许输入其他参数如求解算法选择、误差等。
- 自行实现稀疏矩阵的加、减、乘、转置等运算和线性方程求解算法，禁止调用文件读写、数据存储之外的模块参与计算。
- 注意分析算法的时间、空间复杂度，判断是否适用大规模稀疏线性方程组。

线路设计问题

输入

给定一个图网络$G=(V,E)$，$V=\{1,2,3,\ldots,n\}$为顶点集，表示一些结点；E为有向边集，表示可选的路径。

- V中有一个元素为源点，记作s，源点能提供$o(s)$单位的资源
- $V\backslash\{s\}$中有若干元素为汇点，记作$t_1,t_2,\ldots,t_k$，$k\leq n-1$，每个汇点需要$r(t_i)$个单位的资源
- V中除源点和汇点之外的点不提供也不需要资源

线路设计问题

- 对有向边集E中的任一有向边$a \in E$，a从V中一元素（称为a的起点）指向另一元素（称为a的终点）
- 对任意的有向边$a \in E$
    - 具有容量$u(a)$，经过a的资源不得超过$u(a)$个单位；
    - 具有单位材料成本$c(a)$，经过a的每单位资源产生$c(a)$的材料成本；
    - 具有施工成本$c_0(a)$，若a上有资源经过，则产生$c_0(a)$的施工成本

作业要求

- 寻找G上的设计合理、材料及施工成本最少的资源传输路径，要求得到每条有向边上运输的资源量。
- 对问题适当转化，设计相应的优化算法。设计线路合理性评判标准，并且实现的优化算法能够输出设计合理线路。
- 本作业提供一个问题实例的数据，算法程序应至少在该实例上进行测试；鼓励构造更多实例，充分测试。
- 允许调用现有求解器，或手动实现求解算法。

随机梯度类算法

要求在大规模机器学习问题中实现下述任意一种算法。

本课程《*Optimization Methods for Machine Learning*》章节中将讲授多种随机梯度类算法，包括Stochastic Gradient(P367)，Stochastic Variance Reduced Gradient(P396)，Stochastic Average of Gradient Aggregation(P400)，Subsampled Hessian-Free Inexact Newton(P412)，Stochastic Quasi-Newton Framework(P423)等。

基于动量的优化算法

在本课程讲授内容之外，深度学习领域还常使用基于动量的优化算法，以下进行简单介绍：

- SGD with heavy ball/Polyak momentum:
  动量SGD，可认为是在梯度上做一个移动平均
  $v_k = mv_{k-1} - \alpha g(w_k, \xi_k)$
  $w_{k+1} = w_k + v_k$
  其中$m \in [0, 1)$为momentum parameter

基于动量的优化算法

- SGD with Nesterov momentum:
  Nesterov计算“超前梯度”更新冲量，较之Polyak momentum速度更快
  $v_k = mv_{k-1} - \alpha g(w_k + mv_{k-1}, \xi_k)$
  $w_{k+1} = w_k + v_k$
- Adaptive Gradient:
  AdaGrad算法在更新参数时对不同的参数使用不同的学习率
  $G_k = G_{k-1} + g(w_k, \xi_k)^2$
  $w_{k+1} = w_k - \frac{\alpha}{\sqrt{G_k + \epsilon}} g_k$
  此外Adadelta、RMSprop为AdaGrad算法的改进版本，在此不赘述

基于动量的优化算法

- Adam:
  Adam算法主要贡献在于使用动量和自适应学习率来加快收敛速度
  $m_k = \beta_1 m_{k-1} + (1-\beta_1) g(w_k, \xi_k)$
  $v_k = \beta_2 v_{k-1} + (1-\beta_2) g(w_k, \xi_k)^2$
  $\hat{m}_k = \frac{m_k}{1-\beta_1^k}$
  $\hat{v}_k = \frac{v_k}{1-\beta_2^k}$
  $w_{k+1} = w_k - \frac{\alpha\sqrt{\hat{m}_k}}{\sqrt{\hat{v}_k}+\epsilon}$

大规模机器学习模型

目前主流的大规模机器学习模型大多基于深度学习与神经网络，目前主流深度学习框架是PyTorch、TensorFlow，此外NumPy是最常用的高效数组运算框架（示例代码中以这三者为例给出了一些基本操作实现）。

以下将简单介绍神经网络中的一些基本概念。

多层前馈神经网络

多层前馈神经网络是最基础的神经网络模型，输入层接受外界输入，隐含层与输出层神经元对信号进行加工，每层神经元与下一层神经元全互联，神经元之间不存在同层连接也不存在跨层连接，最终结果由输出层神经元输出。

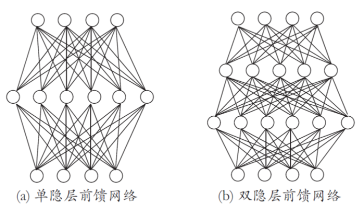

Figure: 多层前馈神经网络

激活函数

激活函数常接于每层网络后，使得网络具有对非线性函数的拟合能力。理想激活函数是阶跃函数，0表示抑制神经元而1表示激活神经元。阶跃函数具有不连续、不光滑等不好的性质，常用的是Sigmoid函数。

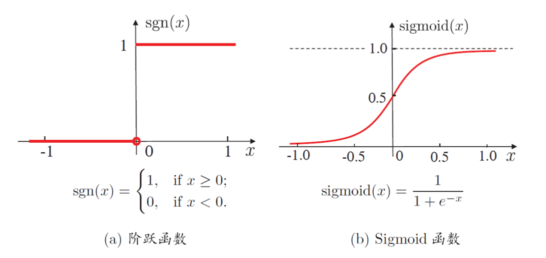

Figure: 激活函数

误差逆传播算法

误差逆传播算法(Error BackPropagation, BP)是最成功的训练多层前馈神经网络的学习算法，其基本步骤如下：

```
输入：训练集 D = {(x_k, y_k)}_{k=1}^m;
      学习率 η.
过程:
在(0,1)范围内随机初始化网络中所有连接权和阈值
repeat
  for all (x_k, y_k) ∈ D do
    根据当前参数和式(5.3) 计算当前样本的输出 ŷ_k;
    根据式(5.10) 计算输出层神经元的梯度项 g_j;
    根据式(5.15) 计算隐层神经元的梯度项 e_h;
    根据式(5.11)-(5.14) 更新连接权 w_hj, v_ih 与阈值 θ_j, γ_h
  end for
until 达到停止条件
输出：连接权与阈值确定的多层前馈神经网络
```

Figure: BP算法流程图

卷积神经网络

目前神经网络模型在深度学习领域应用最广，以图像分类领域为例，卷积神经网络(CNN)在MNIST、CIFAR10等数据集上的表现都远超传统机器学习模型。下简单介绍CNN的特征与结构。

卷积神经网络

CNN使用权共享策略，复合多个卷积层和采样层对输入信号进行加工，在连接层实现与输出目标之间的映射。结构上可分为如下三部分：

- 卷积层：每个卷积层包含多个特征映射，每个特征映射是一个由多个神经元构成的“平面”，通过一种卷积滤波器提取的一种特征
- 采样层：亦称“汇合层”，其作用是基于局部相关性原理进行亚采样，从而在减少数据量的同时保留有用信息
- 连接层：每个神经元被全连接到上一层每个神经元，本质就是传统的神经网络，其目的是通过连接层和输出层的连接完成识别任务

在实际训练时CNN可以用BP进行训练，但在训练中每一组神经元都是用相同的连接权，从而大幅减少了需要训练的参数数目。

作业要求

针对大规模机器学习模型，实现一种随机梯度类算法。

- 由于考察的是相关优化算法在大规模机器学习模型上的有效性，因此要求求解问题时需使用至少3层的神经网络模型，数据集规模n需$\geq 10^3$，网络参数数量p需$\geq 10^3$（过参数化）。
- 程序需实现神经网络模型在所给数据集上的训练+测试过程，具体来说至少需包括如下7个模块：数据集的生成/读取、模型定义、参数初始化、损失函数定义、优化算法定义、训练过程、测试过程，其中优化算法部分需要手动实现，其余部分可直接调用深度学习框架中的相应模块。

作业要求

- 训练过程中的反向传播建议直接调用相应框架中的梯度计算，例如：
  PyTorch：$loss.backward()$
  TensorFlow：$dw, db = g.gradient(loss, [w, b])$
- 若使用非公开数据集，提交时请至少附带10个样例一并打包提交。

Thanks for your attention!