机器学习中的数学第二期第1课:微分学与梯度下降法

七月在线 管老师

2018年2月

主要内容

本课主要内容

介绍微分学基础及其在机器学习中的应用

1. 简介：数学在机器学习中的应用
2. 微分学基本思想和方法
   a) 微分学的核心思想：函数逼近
   b) 微积分的基础语言：极限论
   c) 微分学的基本手法：求导数
   d) 从线性逼近到多项式逼近：泰勒级数
   e) 从低维到高维：多元函数的梯度
3. 梯度下降法和牛顿法
   a) 随机梯度下降
   b) 随机梯度下降的问题与挑战
   c) 随机梯度下降的优化算法选讲

简介：数学在机器学习中的应用

模型建立与选择：对工程问题进行抽象和量化

- 涉及数学知识：综合运用微积分，线性代数，概率统计以及组合数学的知识
- 例如：
  - 各类深度模型中的网络结构与损失函数
  - 支持向量机中的向量空间与度量

模型训练：

- 优化算法：高效稳定的对各类损失函数求极值
- 涉及数学知识：微积分以及优化理论

简介：数学在机器学习中的应用

本系列课程目标：

- 介绍机器学习中所需基础数学知识以及应用，为更进一步的学习打好基础

系列课程涵盖内容：

- 微积分
- 概率统计
- 线性代数
- 优化
- 机器学习实战

微分学核心思想：函数逼近

逼近是人类探讨复杂问题时经常使用的一种手段，比如：

- 人均GDP：使用常数函数来逼近收入分布函数
- 平均速度：使用线性函数来逼近实际运动轨迹
- 年化收益率：使用指数函数来逼近收益函数

微分学核心思想：函数逼近

微分学的核心思想是用**熟悉且简单**的函数对复杂函数进行**局部**逼近

常用作逼近的简单函数包括：

- 线性函数：函数的一阶导数
- 多项式函数：泰勒级数

极限的表述方式：

- 自然语言：当x趋向于a时，$f(x)$ 的极限是L。

- 数学符号：$\lim_{x \to a} f(x) = L$

- 标准语言：对于任意的$\epsilon > 0$, 存在一个$\delta > 0$，使得对于任何的$x_* \in (a - \delta, a + \delta)$, 都有，$| f(x_*) - L | < \epsilon$

微分学基础语言：极限论

无穷小：

一般把趋于零的极限称为无穷小。

无穷小阶数：

趋于零的速度越快的无穷小，其阶数越高。比 $x^n, x \to 0$, 趋于零速度还快的无穷小记为$o(x^n)$.

两边夹定理:

如果$f(x) < g(x) < h(x)$,而且这三个函数都在a点处有极限，那么

$$\lim_{x \to a} f(x) \le \lim_{x \to a} g(x) \le \lim_{x \to a} h(x)$$

微分学基础语言：极限论

重要极限: (两边夹定理应用)

- 三角函数：$\lim\limits_{x\to 0}\frac{\sin(x)}{x}=1$
- 自然对数底数：$e=\lim\limits_{n\to\infty}\left(1+\frac{1}{n}\right)^n$
- 指数函数：$\lim\limits_{x\to 0}\frac{e^x-1}{x}=1$

微分学基本手法：求导数

微分学的核心思想是逼近：

一阶导数

$$f'(x) = \lim_{\Delta \to 0} \frac{f(x+\Delta) - f(x)}{\Delta}$$

- 几何意义：用直线逼近曲线
- 代数意义：用线性函数逼近复杂函数

微分学基本手法：求导数

对函数进行线性逼近：

假设函数$f(x)$是一个可微函数，x_0是定义域中的一个点，那么

$$f(x_0+\Delta)=f(x_0)+\Delta\cdot\frac{d}{dx}f(x_o)+o(\Delta)$$

注释：现代的数学也常常使用如下记号:

$$dx=\Delta$$

$$df(x_0)=f(x_0+\Delta)-f(x_0)+o(\Delta)$$

那么上面的微分公式就写成：

$$df(x_0)=\frac{d}{dx}f(x_0)\cdot dx$$

微分学基本手法：求导数

常见函数的导数：

- 多项式函数：
  - $\frac{\mathrm{d}}{\mathrm{dx}} x^n = n \cdot x^{n-1}$
- 三角函数：
  - $\frac{\mathrm{d}}{\mathrm{dx}} \sin(x) = \cos(x)$
- 指数函数：
  - $\frac{\mathrm{d}}{\mathrm{dx}} e^x = e^x$

从线性逼近到多项式逼近：泰勒级数

一元微分学的顶峰：Taylor 级数(对函数进行高阶逼近)

导数的导数就是二阶导数

n阶导数的导数就是$n+1$阶导数

Taylor级数就是利用n阶导数来对函数进行高阶逼近。

从线性逼近到多项式逼近：泰勒级数

Taylor 级数(对函数进行高阶逼近)

如果一个函数$f(x)$ 是n阶可微函数，那么：

$$f(x_0+\Delta)=f(x_0)+f'(x_0)\cdot\Delta+\frac{f^{(2)}(\mathrm{x}_0)}{2}\cdot\Delta^2+\cdots+\frac{f^{(\mathrm{n})}(\mathrm{x}_0)}{n!}\cdot\Delta^n+o(\Delta^n)$$

函数 $f(x)$的n阶Taylor级数就是与$f(x)$拥有相同前n阶导数的n阶多项式。

从线性逼近到多项式逼近：泰勒级数

当 n=2时，Taylor 级数就成为一个二次逼近：

对于2阶可微函数$f(x)$：

$$f(x_0+\Delta)=f(x_0)+f'(x_0)\cdot\Delta+\frac{f^{(2)}(x_0)}{2}\cdot\Delta^2+o(\Delta^2)$$

而这就构成了牛顿法的基础

从低维到高维：多元函数的梯度

对于一个二元函数$f(x,y)$,偏导数定义为:

$$\partial_x f(x,y) = \frac{\partial}{\partial x} f(x,y) = \lim_{\Delta_x \to 0} \frac{f(x+\Delta_x, y) - f(x,y)}{\Delta_x}$$

$$\partial_y f(x,y) = \frac{\partial}{\partial y} f(x,y) = \lim_{\Delta_y \to 0} \frac{f\left(x, y+\Delta_y\right) - f(x,y)}{\Delta_y}$$

沿着 方向 $v = (a,b)$的方向导数为

$$\nabla_v f(x,y) = \lim_{\Delta \to 0} \frac{f(x+\Delta\cdot a, y+\Delta\cdot b) - f(x,y)}{\Delta}$$

偏导数就是沿着坐标轴方向的方向导数

从低维到高维：多元函数的梯度

多元可微函数，一个二元函数$f(x,y)$一阶可微，如果存在L_x, L_y 使得：

$$f\left(x+\Delta_x, y+\Delta_y\right) = f(x,y) + L_x \cdot \Delta_x + L_y \cdot \Delta_y + o(|\Delta_x| + |\Delta_y|)$$

从低维到高维：多元函数的梯度

梯度 (以二元函数为例):

对于一个可微函数$f(x,y)$,梯度定义为$\nabla f(x,y)=\left(\partial_x f,\partial_y f\right)^T$

梯度的代数意义：

其任意方向的偏导数可以由梯度来表示，如果$v=(a,b)$,

$$\nabla_v f(x,y)=v\cdot\nabla f(x,y)=a\partial_x f(x,y)+b\partial_y f(x,y)$$

梯度的几何意义：梯度方向就是函数增长最快的方向

梯度下降法和牛顿法

梯度下降法：

如果J(θ)是一个多元函数，在θ_0点附近对J(θ)做线性逼近

$$J(\theta_0 + \Delta_\theta) = J(\theta_0) + \Delta_\theta^T \cdot \nabla J(\theta_0) + o(|\Delta_\theta|)$$

这个线性逼近不能告诉我们极值点在什么地方，他只能告诉我们极值点在什么方向. 所以我们只能选取一个比较“小”的学习率η来沿着这个方向走下去，并得到梯度下降法的序列：

$$\theta_n = \theta_{n-1} - \eta_{n-1} \nabla J(\theta_{n-1})$$

梯度下降法和牛顿法

梯度下降法：对函数进行一阶逼近寻找函数下降最快的方向

牛顿法：就是对函数的二阶逼近，并直接估计函数的极小值点

$$f(\theta_0+\Delta_\theta)=f(\theta_0)+\Delta_\theta^T\cdot\nabla J(\theta_0)+\frac{1}{2}\Delta_\theta^T HJ(\theta_0)\Delta_\theta+\mathrm{o}(\,|\Delta_\theta|^2)$$

于是关于Δ_θ的剃度为：

$$\nabla J(\theta_0)+HJ(\theta_0)\Delta_\theta+\mathrm{o}(\,|\Delta_\theta|\,)$$

零点的近似值为：

$$\Delta_\theta=HJ(\theta_0)^{-1}\cdot\nabla J(\theta_0)$$

梯度下降法和牛顿法

困难：

1. **梯度的计算**

   在机器学习和统计参数估计问题中目标函数经常是求和函数的形式

   $$J_X(\theta) = \sum_i J_{x_i}(\theta)$$

   其中每一个函数$J_{x_i}(\theta)$都对应于一个样本x_i。

   - **当样本量极大时，梯度的计算就变得非常耗时耗力。**

2. **学习率的选择**

   学习率选择过小会导致算法收敛太慢，学习率选择过大容易导致算法不收敛。

   - **如何选择学习率需要 具体问题具体分析**

随机梯度下降法

随机梯度下降法主要为了解决第一个问题：梯度计算

由于随机梯度下降法的引入，我们通常将梯度下降法分为三种类型：

1. **批梯度下降法(GD)**
   原始的梯度下降法
2. **随机梯度下降法(SGD)**
   每次梯度计算只使用一个样本
   - 避免在类似样本上计算梯度造成的冗余计算
   - 增加了跳出当前的局部最小值的潜力
   - 在逐渐缩小学习率的情况下，有与批梯度下降法类似的收敛速度
3. **小批量随机梯度下降法(Mini Batch SGD)**
   每次梯度计算使用一个小批量样本
   - 梯度计算比单样本更加稳定
   - 可以很好的利用现成的高度优化的矩阵运算工具

注意：神经网络训练的文献中经常把 Mini Batch SGD 称为 SGD

随机梯度下降法的困难

随机梯度下降法的主要困难在于前述的第二个问题：学习率的选取

1. **局部梯度的反方向不一定是函数整体下降的方向**
    - 对图像比较崎岖的函数，尤其是隧道型曲面，梯度下降表现不佳
2. **预定学习率衰减法的问题**
    - 学习率衰减法很难根据当前数据进行自适应
3. **对不同参数采取不同的学习率的问题**
    - 在数据有一定稀疏性时，希望对不同特征采取不同的学习率
4. **神经网络训练中梯度下降法容易被困在鞍点附近的问题**
    - 比起局部极小值，鞍点更加可怕

随机梯度下降法的优化算法

为什么不用牛顿法？

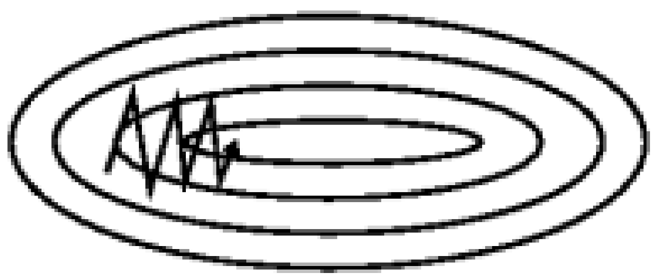
- 牛顿法要求计算目标函数的二阶导数(Hessian matrix)，在高维特征情形下这个**矩阵非常巨大**，计算和存储都成问题
- 在使用小批量情形下，牛顿法对于二阶导数的估计**噪音太大**
- 在目标函数非凸时，牛顿法更容易收到**鞍点**甚至**最大值点**的吸引

随机梯度下降法的优化算法

动量法（Momentum）（适用于隧道型曲面）

梯度下降法在狭长的隧道型函数上表现不佳，如下图所示

- 函数主体缓缓向右方下降
- 在主体方向两侧各有一面高墙，导致垂直于主体方向有更大的梯度
- 梯度下降法会在隧道两侧频繁震荡。



(a) SGD without momentum

随机梯度下降法的优化算法

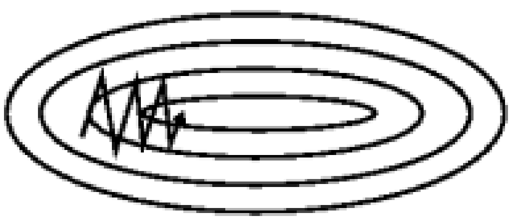
动量法（Momentum）（适用于隧道型曲面）

动量法每次更新都吸收一部分上次更新的余势：

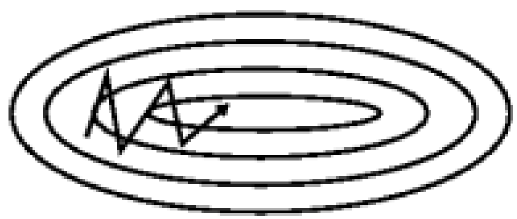
$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$
$$\theta_t = \theta_{t-1} - v_t$$

这样主体方向的更新就得到了更大的保留，从而效果被不断放大。

物理上这就像是推一个很重的铁球下山，因为铁球保持了下山主体方向的动量，所以在隧道上沿两侧震荡测次数就会越来越少。



(a) SGD without momentum



(b) SGD with momentum

随机梯度下降法的优化算法

Nesterov accelerated gradient（动量法的改进算法）

动量法的一个问题在于：从山顶推下的铁球会越滚越快，以至于到了山底停不下来。我们希望算法更加聪明一些，可以在到达底部之前就自己刹车。

利用主题下降方向提供的先见之明，预判自己下一步的位置，并到预判位置计算梯度。

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta - \gamma v_{t-1})$$
$$\theta = \theta - v_t$$

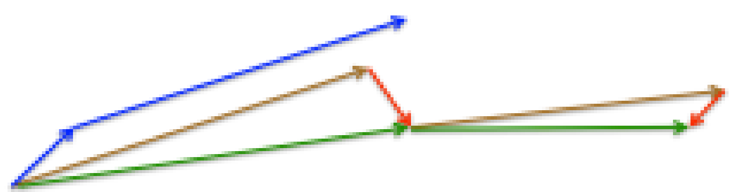


Figure 3: Nesterov update (Source: G. Hinton's lecture 6c)

随机梯度下降法的优化算法

Adagrad （自动调整学习率，适用于稀疏数据）

梯度下降法在每一步对每一个参数使用相同的学习率，这种一刀切的做法不能有效的利用每一个数据集自身的特点。

Adagrad 是一种自动调整学习率的方法

- 随着模型的训练，学习率自动衰减
- 对于更新频繁的参数，采取较小的学习率
- 对于更新不频繁的参数，采取较大的学习率

随机梯度下降法的优化算法

Adagrad （自动调整学习率，适用于稀疏数据）

为了实现对于更新频繁的参数使用较小的学习率，Adagrad 对每个参数历史上的每次更新进行叠加，并以此来做下一次更新的惩罚系数。

梯度: $g_{t,i} = \nabla_\theta J(\theta_i)$

梯度历史矩阵： G_t 对角矩阵，其中$G_{t,ii} = \sum_k g_{k,i}^2$

参数更新：

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i}$$

随机梯度下降法的优化算法

Adadelta （Adagrad 的改进算法）

- Adagrad 的一个问题在于随着训练的进行，学习率快速单调衰减。

Adadelta 则使用梯度平方的移动平均来取代全部历史平方和。

定义移动平均： $E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$

于是就得到参数更新法则：

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{E[g^2]_{t,ii} + \epsilon}} \cdot g_{t,i}$$

随机梯度下降法的优化算法

Adadelta （ Adagrad 的改进算法 ）

- Adagrad 以及一般的梯度下降法的另一个问题在于，梯度与参数的单位不匹配。

Adadelta 使用参数更新的移动平均来取代学习率η. 于是参数更新法则：

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\sqrt{\mathrm{E}[\Delta\theta]_{t-1}}}{\sqrt{E[g^2]_{t,ii} + \epsilon}} \cdot g_{t,i}$$

注意： Adadelta 的第一个版本也叫做 RMSprop，是Geoff Hinton独立于Adadelta提出来的。

随机梯度下降法的优化算法

Adam （结合了动量法与Adamdelta的算法）

如果把Adadelta里面梯度的平方和看成是梯度的二阶矩，那么梯度自身的求和就是一阶矩。Adam 算法在Adadelta的二阶矩基础之上又引入了一阶矩。

而一阶矩，其实就类似于动量法里面的动量。

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

随机梯度下降法的优化算法

Adam （结合了动量法与Adamdelta的算法）

如果把Adadelta里面梯度的平方和看成是梯度的二阶矩，那么梯度自身的求和就是一阶矩。Adam 算法在Adadelta的二阶矩基础之上又引入了一阶矩。

而一阶矩，其实就类似于动量法里面的动量。

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

于是参数更新法则为：

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon} m_t$$

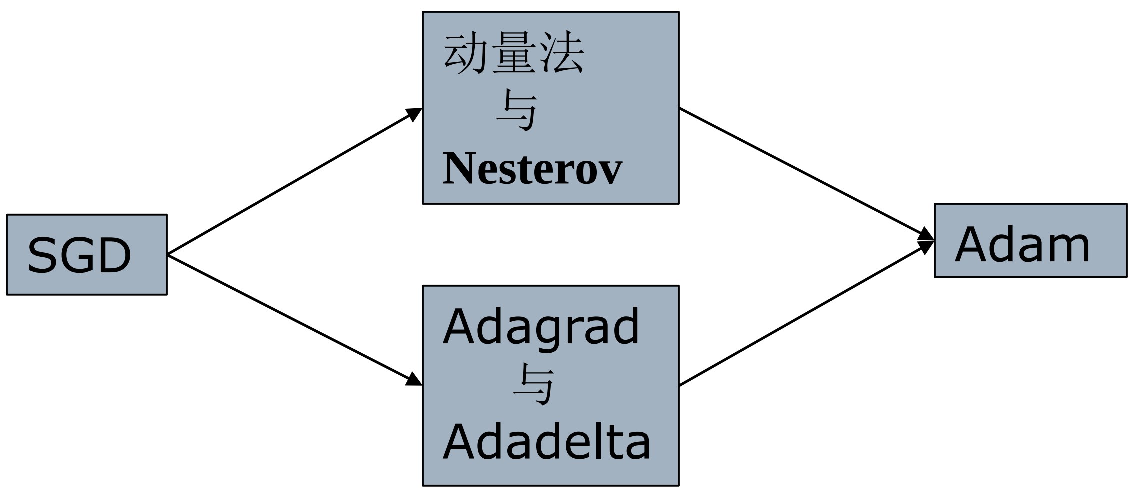
注意：实际操作中 v_t与m_t采取了更好的无偏估计，来避免前几次更新时候数据不足的问题。

随机梯度下降法的优化算法

究竟如何选择算法呢?

- 动量法与Nesterov的改进方法着重解决目标函数图像崎岖的问题
- Adagrad与Adadelta主要解决学习率更新的问题
- Adam集中了前述两种做法的主要优点

目前为止 Adam **可能**是几种算法中综合表现最好的



作业

参考资料：

- 数学分析教程,常庚哲,史济怀
- 简明微积分,龚升
- 微积分讲义,陈省身
- 深度学习 前四章：https://www.jiqizhixin.com/articles/2017-04-14-6
- 计算机科学中的数学(MIT&GOOGLE)：https://courses.csail.mit.edu/6.042/spring17/mcs.pdf

作业

- 作业,数学分析教程,常庚哲,史济怀 (p142:2,3,7,8; p143:3,4,6; p148:2,3,6; p176:8,11; p210:4,5; p211:6)