

# 机器学习笔记

Notes on Machine Learning

J.R.Tsien

[jade.ray.tsien@gmail.com](mailto:jade.ray.tsien@gmail.com)

# 目 录

第1章 绪论 .....	1
1.1 最优化问题 .....	1
1.1.1 原始问题 .....	1
1.1.2 拉格朗日对偶问题 .....	1
1.2 梯度下降法 (gradient descent) .....	1
1.3 牛顿法和拟牛顿法 .....	1
1.3.1 牛顿法 (Newton Method) .....	2
1.3.2 拟牛顿法 (Quasi Newton Method) .....	2
1.3.2.1 拟牛顿条件 .....	2
1.3.2.2 DFP算法 .....	3
1.4 矩阵分析 (matrix analysis) .....	3
1.4.1 迹(Trace)和导数(matrix derivatives) .....	3
1.5 泛函分析 .....	5
1.6 常用不等式 .....	5
1.6.1 柯西不等式 (Cauchy Inequality) .....	5
1.6.2 赫尔德不等式 (Hölder Inequality) .....	5
1.6.3 闵可夫斯基不等式 (Minkowski Inequality) .....	5
第2章 回归分析 .....	6
2.1 线性回归 .....	6
2.1.1 直接求解 .....	6

---

2.1.2 牛顿法 (Newton's Method) .....	7
2.1.3 批处理梯度下降法 (batch gradient descent) .....	8
2.1.4 随机梯度下降法 (stochastic gradient descent) .....	8
2.2 局部加权线性回归 (LWR) .....	8
2.3 逻辑斯蒂回归 .....	9
第3章 K近邻 .....	10
3.1 模型 .....	10
3.2 策略 .....	10
3.3 算法 .....	10
第4章 朴素贝叶斯法 .....	11
第5章 决策树 .....	12
第6章 逻辑斯谛回归和最大熵 .....	13
第7章 支持向量机 .....	14
第8章 提升方法 .....	15
第9章 EM方法 .....	16
9.1 Jensen不等式 .....	16
第10章 隐马尔可夫模型 .....	17
参考文献 .....	18

# 第1章 绪论

## § 1.1 最优化问题

### 1.1.1 原始问题

### 1.1.2 拉格朗日对偶问题

## § 1.2 梯度下降法 (gradient descent)

梯度下降法或者最速下降法 (steepest descent) 是求解无约束最优化问题的方法。特点是实现起来比较简单。其原理是如果函数  $f(x)$  在点  $a$  处可微且有定义，那么函数  $f(x)$  在  $a$  点沿着梯度的反方向，即  $-\nabla f(a)$ ，下降最快。

所以，可以从一个初始值  $x_0$  出发，沿梯度反方向迭代的更新解。如下

$$x_{n+1} = x_n - \alpha \nabla f(x_n)$$

直到  $x_n$  的值不再发生变化，或者变化很小，此时， $x_n$  等于或者接近  $f(x)$  的极小值。 $\alpha$  称为学习率 (learning rate)。 $\alpha$  值过大，可能会在最小值附近振荡。 $\alpha$  值过小，可能学习的时间比较长。同时， $\alpha$  值的选取可以是预先设定的固定值，也可以是根据解更新的情况变化的值。

梯度下降法的一个问题在于，能否得到最优解取决于初始值的选取。

## § 1.3 牛顿法和拟牛顿法

### 1.3.1 牛顿法 (Newton Method)

牛顿法，或牛顿-拉夫逊法 (Newton-Raphson Method) 也是求解无约束优化问题的常用方法。牛顿法是二阶收敛的算法（不仅考虑梯度方向，同时考虑梯度的梯度），而梯度下降法是一阶收敛的，因此牛顿法的收敛速度比梯度下降快。换句话说，牛顿法用二次曲面来拟合当前所在位置的局部曲面，然后按照曲率最大的方向下降。而梯度法是用一个平面去拟合局部曲面，然后按照平面的法向量的方向下降。但是一次迭代的代价比较高，因为需要计算矩阵的逆。

考虑无约束的最优化问题

$$\min_{x \in \mathbb{R}^n} f(x)$$

假设 $f(x)$ 有二阶连续偏导数，且设第 $k$ 次迭代的解为 $x_k$ ，将 $f(x)$ 在点 $x_k$ 处进行二阶泰勒展开

$$f(x) = f(x_k) + (x - x_k)f'(x_k) + \frac{1}{2}(x - x_k)^2 f''(x_k)$$

函数 $f(x)$ 在下次迭代点 $x_{k+1}$ 处取得极值的必要条件是 $f'(x_{k+1}) = 0$ ，即

$$f'(x)|_{x_{k+1}} = f'(x_k) + (x_{k+1} - x_k)f''(x_k) = 0$$

解上式得到

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

迭代停止的条件可以设定为 $f'(x_k) < \epsilon$ 。

当 $x$ 是向量的时候，其一阶导数要修改成梯度的形式，二阶导数修改成其Hessian矩阵，即

$$f(x) = f(x_k) + (x - x_k)\nabla_x f(x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k)$$

当 $H(x_k)$ 是正定矩阵时， $f(x)$ 的极值为极小值。其更新公式是

$$x_{k+1} = x_k - H^{-1}(x_k)\nabla_x f(x_k)$$

迭代终止的条件 $\nabla_x f(x_k) < \epsilon$ 。

### 1.3.2 拟牛顿法 (Quasi Newton Method)

牛顿法中Hessian矩阵求逆比较复杂，所以实际中会考虑用一个近似的正定对称矩阵来代替Hessian矩阵，这就是拟牛顿法。不同的替代方法形成了不同的拟牛顿法。

#### 1.3.2.1 拟牛顿条件

首先将函数 $f(x)$ 在第 $k$ 次迭代点处泰勒展开

$$f(x) = f(x_k) + (x - x_k)\nabla_x f(x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k)$$

求其此时的梯度

$$\nabla f(x) = \nabla f(x_k) + (x - x_k)H(x_k)$$

令  $x = x_{k+1}$ ,

$$\nabla f(x_{k+1}) - \nabla f(x_k) = (x_{k+1} - x_k)H(x_k)$$

左边是梯度的变化, 右边  $(x_{k+1} - x_k)$  是自变量的变化, 分别记为

$$g_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

$$\delta_k = x_{k+1} - x_k$$

得到拟牛顿条件

$$g_k = H_k \delta_k \tag{1.1}$$

或

$$H_k^{-1} g_k = \delta_k \tag{1.2}$$

拟牛顿法选择  $G_k \approx H_k^{-1}$ , 或者  $B_k \approx H_k$ 。而且, 尽量使得近似矩阵的更新方式为迭代更新

$$G_{k+1} = G_k + \Delta G_k$$

一般令  $G_0 = I$  为单位对角阵, 所以只需要找到  $\Delta G_k$  即可。

### 1.3.2.2 DFP算法

## § 1.4 矩阵分析 (matrix analysis)

### 1.4.1 迹(Trace)和导数(matrix derivatives)

令  $f: \mathbb{R}^{m \times n} \mapsto \mathbb{R}$  表示将  $m \times n$  ( $m$ -by- $n$ ) 矩阵映射为实数的函数。定义  $f$  对矩阵  $\mathbf{A}$  的导数

$$\nabla_{\mathbf{A}} f(\mathbf{A}) = \begin{pmatrix} \frac{\partial f(\mathbf{A})}{\partial a_{11}} & \cdots & \frac{\partial f(\mathbf{A})}{\partial a_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f(\mathbf{A})}{\partial a_{n1}} & \cdots & \frac{\partial f(\mathbf{A})}{\partial a_{nn}} \end{pmatrix}$$

矩阵的迹 (trace) 表示的是矩阵的对角元素的和,

$$tr \mathbf{A} = \sum_{i=1}^n a_{ii}$$

假设  $A, B, C, D$  均是方阵

$$\text{tr}ABCD = \text{tr}DABC = \text{tr}CDAB = \text{tr}BCDA \quad (1.3)$$

循环将最右边矩阵放到最左边。假设  $a$  是实数

$$\text{tr}A = \text{tr}A^T \quad (1.4)$$

$$\text{tr}(A + B) = \text{tr}A + \text{tr}B \quad (1.5)$$

$$\text{tr}aA = a\text{tr}A \quad (1.6)$$

下面的一些公式出自 Andrew Ng 的机器学习讲义，这里证明一下。

$$\nabla_A \text{tr}AB = B^T \quad (1.7)$$

$$\nabla_{A^T} f(A) = (\nabla_A f(A))^T \quad (1.8)$$

$$\nabla_A \text{tr}ABA^T C = CAB + C^T AB^T \quad (1.9)$$

$$\nabla_A |A| = |A|(A^{-1})^T \quad (1.10)$$

**证明** (1)  $(\nabla_A \text{tr}AB)_{ij} = \frac{\partial \text{tr}AB}{\partial a_{ij}} = \frac{\partial \sum_m \sum_k a_{mk} b_{km}}{\partial a_{ij}}$ ，只有当  $m = i, k = j$  时才有  $a_{ij}$  的系数，所以  $(\nabla_A \text{tr}AB)_{ij} = b_{ji}$ ，即证。

(2)  $(\nabla_{A^T} f(A))_{ij} = \frac{\partial f(A)}{\partial a_{ji}}$ ，即证。

(3)  $\text{tr}ABA^T C = \sum_m \sum_k \sum_t \sum_s a_{mk} b_{kt} a_{st} c_{sm}$ ，所以

$$\begin{aligned} (\nabla_A \text{tr}ABA^T C)_{ij} &= \frac{\partial \sum_m \sum_k \sum_t \sum_s a_{mk} b_{kt} a_{st} c_{sm}}{\partial a_{ij}} \\ &= \sum_m \sum_k \sum_t \sum_s \frac{\partial a_{mk}}{\partial a_{ij}} b_{kt} a_{st} c_{sm} + \sum_m \sum_k \sum_t \sum_s a_{mk} b_{kt} \frac{\partial a_{st}}{\partial a_{ij}} c_{sm} \end{aligned}$$

左边，令  $m = i, k = j$ ，右边，令  $s = i, t = j$ ，

$$\begin{aligned} (\nabla_A \text{tr}ABA^T C)_{ij} &= \sum_t \sum_s b_{jt} a_{st} c_{si} + \sum_m \sum_k a_{mk} b_{kj} c_{im} \\ &= \sum_t \sum_s b_{jt} a_{st} c_{si} + \sum_m \sum_k c_{im} a_{mk} b_{kj} \\ &= (BA^T C)_{ji} + (CAB)_{ij} \\ &= (C^T AB^T + CAB)_{ij} \end{aligned}$$

## § 1.5 泛函分析

### § 1.6 常用不等式

#### 1.6.1 柯西不等式 (Cauchy Inequality)

柯西不等式, 又称柯西-施瓦茨不等式 (Cauchy-Schwarz inequality)。对于一个内积空间所有向量  $\mathbf{x}$  和  $\mathbf{y}$ ,

$$|\langle \mathbf{x}, \mathbf{y} \rangle|^2 \leq \langle \mathbf{x}, \mathbf{x} \rangle \cdot \langle \mathbf{y}, \mathbf{y} \rangle$$

其中  $\langle \cdot, \cdot \rangle$  表示内积 (点积), 当且仅当  $\mathbf{x}$  与  $\mathbf{y}$  线性相关时等式成立。

对于欧几里得空间  $\mathbb{R}^2$ ,

$$\left( \sum_{i=1}^n x_i y_i \right)^2 \leq \left( \sum_{i=1}^n x_i^2 \right) \left( \sum_{i=1}^n y_i^2 \right)$$

当且仅当  $\frac{x_1}{y_1} = \frac{x_2}{y_2} = \dots = \frac{x_n}{y_n}$  时等式成立。

#### 1.6.2 赫尔德不等式 (Hölder Inequality)

赫尔德不等式揭示了  $L^p$  空间的相互关系。设  $S$  为测度空间,  $1 \leq p, q \leq \infty$ , 且  $\frac{1}{p} + \frac{1}{q} = 1$ , 若  $f \in L^p(S)$ ,  $g \in L^q(S)$ , 则  $fg \in L^1(S)$ , 且

$$\|fg\|_1 \leq \|f\|_p \|g\|_q$$

写成序列或向量的形式

$$\sum_{i=1}^n |a_i b_i| \leq \left( \sum_{i=1}^n |a_i|^p \right)^{\frac{1}{p}} \left( \sum_{i=1}^n |b_i|^q \right)^{\frac{1}{q}}$$

#### 1.6.3 闵可夫斯基不等式 (Minkowski Inequality)

闵可夫斯基不等式表明  $L^p$  空间是一个赋范向量空间。设  $S$  是一个度量空间,  $f, g \in L^p(S)$ ,  $1 \leq p \leq \infty$ , 那么  $f + g \in L^p(S)$ , 有

$$\|f + g\|_p \leq \|f\|_p + \|g\|_p$$

写成序列或向量的形式

$$\left( \sum_{k=1}^n |x_k + y_k|^p \right)^{\frac{1}{p}} \leq \left( \sum_{k=1}^n |x_k|^p \right)^{\frac{1}{p}} + \left( \sum_{k=1}^n |y_k|^p \right)^{\frac{1}{p}}$$



## 第2章 回归分析

给定数据集

$$T = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\},$$

其中,  $x^{(i)} = (x_0, x_1, \dots, x_n) \in \mathcal{X} = \mathbb{R}^{n+1}$ ,  $y^{(i)} \in \mathcal{Y}$ ,  $i = 0, 2, \dots, m$ , 且  $x_0 = 1$  (表示截距, intercept term)。回归分析的任务是找出输入  $x$  与输出  $y$  之间的关系。

### § 2.1 线性回归

假设输入与输出之间满足的关系是线性的,  $\theta$  称为参数 (parameters) 或者权重 (weights)

$$y = h_{\theta}(x) = \theta^T x = \sum_{i=0}^n \theta_i x_i, \quad \theta \in \mathbb{R}^{n+1}$$

对于这个模型, 需要有一个损失函数 (cost function) 来表示其对训练数据的拟合程度

$$J(\theta) = \frac{1}{2} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2.$$

这个被称为最小二乘方效用函数 (least-square cost function)。

则这个问题的求解可以表述为如下的无约束最优化问题

$$\min_{\theta} J(\theta)$$

#### 2.1.1 直接求解

直接求  $J(\theta)$  对  $\theta$  的极值。首先定义设计矩阵 (design matrix)  $X$

$$X = \begin{bmatrix} (x^{(1)})^T & (x^{(2)})^T & \dots & (x^{(m)})^T \end{bmatrix}^T$$

$$Y = \begin{bmatrix} y^{(1)} & y^{(2)} & \dots & y^{(m)} \end{bmatrix}^T$$

则有

$$X\theta - Y = \begin{bmatrix} h_{\theta}(x^{(1)}) - y^{(1)} & \dots & h_{\theta}(x^{(m)}) - y^{(m)} \end{bmatrix}^T$$

考虑到  $z^T z = \sum_i z_i^2$ , 有

$$J(\theta) = \frac{1}{2}(X\theta - Y)^T(X\theta - Y) = \frac{1}{2} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

使用公式  $\nabla_A \text{tr} ABA^T C = CAB + C^T AB^T$ , 且由于  $J(\theta)$  只是个实数, 所以

$$\begin{aligned} \nabla_{\theta} J(\theta) &= \nabla_{\theta} \text{tr} J(\theta) \\ &= X^T X\theta - X^T Y \end{aligned}$$

令  $\nabla_{\theta} J(\theta) = 0$  得到

$$\theta = (X^T X)^{-1} X^T Y$$

不过, 这个公式用来直接计算  $\theta$  不现实, 因为矩阵求逆比较麻烦, 同时可能会是数值不稳定的矩阵。

### 2.1.2 牛顿法 (Newton's Method)

已知最优化问题

$$\min_{\theta} J(\theta) = \frac{1}{2} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

函数  $J(\theta)$  的 Hessian 矩阵是 (注意  $x^{(i)}$  和  $\theta$  都是列向量)

$$H(\theta) = \begin{bmatrix} \frac{\partial^2 J(\theta)}{\partial \theta_1 \partial \theta_1} & \frac{\partial^2 J(\theta)}{\partial \theta_1 \partial \theta_2} & \dots & \frac{\partial^2 J(\theta)}{\partial \theta_1 \partial \theta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J(\theta)}{\partial \theta_n \partial \theta_1} & \frac{\partial^2 J(\theta)}{\partial \theta_n \partial \theta_2} & \dots & \frac{\partial^2 J(\theta)}{\partial \theta_n \partial \theta_n} \end{bmatrix} = \sum_{i=1}^m x^{(i)} (x^{(i)})^T = X^T X$$

又,  $J(\theta) = \frac{1}{2}(X\theta - Y)^T(X\theta - Y)$ , 所以其梯度

$$\nabla_{\theta} J(\theta) = X^T X\theta - X^T Y$$

代入牛顿法的迭代公式

$$\theta_{n+1} = \theta_n - (X^T X)^{-1} (X^T X\theta_n - X^T Y) = (X^T X)^{-1} X^T Y$$

### 2.1.3 批处理梯度下降法 (batch gradient descent)

$J(\theta)$ 对 $\theta$ 的梯度

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})x^{(i)}$$

所以，更新公式为

$$\begin{aligned}\theta_{n+1} &= \theta_n - \alpha \frac{\partial J(\theta)}{\partial \theta} \\ &= \theta_n - \alpha \sum_{i=1}^m (h_{\theta_n}(x^{(i)}) - y^{(i)})x^{(i)}\end{aligned}$$

这个公式更新时每次都需要全部的训练数据集，所以称之为批处理梯度下降法。当数据集比较大时，进行一次更新就比较耗费时间。

### 2.1.4 随机梯度下降法 (stochastic gradient descent)

随机梯度下降法，也称增量梯度下降 (incremental gradient descent)，一次使用一条训练数据来更新参数。算法如下

$$\theta := \theta - \alpha (h_{\theta}(x^{(i)}) - y^{(i)})x^{(i)}, i = 1, 2, \dots, m$$

随机梯度下降法更新的速度比梯度下降法快，但可能收敛不到最优值，不过通过调节学习率可以使得算法得到较好的解。

## § 2.2 局部加权线性回归 (LWR)

线性回归的方法是参数学习算法 (parametric learning algorithm)，其参数的个数，即特征的个数是固定的，一旦算法学习完成，训练数据集就不再对参数产生影响。但是，当选取的参数过多时，可能存在过拟合问题，而当选取的参数过少时，存在欠拟合问题。局部加权线性回归 (local weighted linear regression, LWR) 是一种非参学习算法 (non-parametric learning algorithm)，其参数是随着预测点的不同而发生变化的，每有一个新的预测点，就需要整个训练数据集重新参与学习。所谓局部，是因为目标函数的逼近仅仅根据查询点附近的数据。所谓加权，是因为每个训练样例的贡献都是由它与查询点间的距离加权的。而回归是指数值逼近的方法。

线性回归的优化目标是

$$\min_{\theta} \sum_{i=1}^m (\theta^T x^{(i)} - y^{(i)})^2$$

而LWR的优化目标是在上述公式上增加一个距离乘法项

$$\min_{\theta} \sum_{i=1}^m w^{(i)} (\theta^T x^{(i)} - y^{(i)})^2$$

一个相对标准的权重选择是

$$w^{(i)} = \exp\left(-\frac{(x^{(i)} - x)^2}{2\tau^2}\right)$$

其中， $x$ 是要预测的输入数据， $\tau$ 称为带宽（bandwidth）。权重项使得离输入数据 $x$ 越近的点影响越大。

非参数学习算法在局部预测能力上有时要比参数学习算法好，但是缺点是每次做预测都要重新学习，耗费时间空间。

## § 2.3 逻辑斯蒂回归

## 第3章 K近邻

### § 3.1 模型

### § 3.2 策略

### § 3.3 算法

## 第4章 朴素贝叶斯法

## 第5章 决策树

## 第6章 逻辑斯谛回归和最大熵



## 第7章 支持向量机

## 第8章 提升方法

## 第9章 EM方法

### § 9.1 Jensen不等式

## 第10章 隐马尔可夫模型

## 参考文献

- [1] 李航著. 《统计学习方法》. 北京:清华大学出版社, 2012, 3
- [2] Jiawei Han, Micheline Kamber, Jian Pei 著.范明, 孟小峰译. 《数据挖掘: 概念与技术》. 机械工业出版社, 2012, 8
- [3] Tom M. Mitchell 著.曾华军等译. 《机器学习》.机械工业出版社, 2003, 1
- [4] Andrew Ng, 《Machine Learning》公开课讲义