机器学习经典算法之-----最小二乘法

一.背景

5月9号到北大去听hulu的讲座《推荐系统和计算广告在视频行业应用》，想到能见到传说中的项亮大神，特地拿了本《推荐系统实践》求签名。讲座开始，主讲人先问了下哪些同学有机器学习的背景，我恬不知耻的毅然举手，真是惭愧。后来主讲人在讲座中提到了最小二乘法，说这个是机器学习最基础的算法。神马，最基础，我咋不知道呢！ 看来以后还是要对自己有清晰认识。

回来赶紧上百度，搜了下什么是最小二乘法。

先看下百度百科的介绍：最小二乘法（又称最小平方法）是一种数学优化技术。它通过最小化误差的平方和寻找数据的最佳函数匹配。利用最小二乘法可以简便地求得未知的数据，并使得这些求得的数据与实际数据之间误差的平方和为最小。最小二乘法还可用于曲线拟合。其他一些优化问题也可通过最小化能量或最大化熵用最小二乘法来表达。

通过这段描述可以看出来，最小二乘法也是一种优化方法，求得目标函数的最优值。并且也可以用于曲线拟合，来解决回归问题。难怪《统计学习方法》中提到，回归学习最常用的损失函数是平方损失函数，在此情况下，回归问题可以著名的最小二乘法来解决。看来最小二乘法果然是机器学习领域做有名和有效的算法之一。

二. 最小二乘法

我们以最简单的一元线性模型来解释最小二乘法。什么是一元线性模型呢？ 监督学习中，如果预测的变量是离散的，我们称其为分类（如决策树，支持向量机等），如果预测的变量是连续的，我们称其为回归。回归分析中，如果只包括一个自变量和一个因变量，且二者的关系可用一条直线近似表示，这种回归分析称为一元线性回归分析。如果回归分析中包括两个或两个以上的自变量，且因变量和自变量之间是线性关系，则称为多元线性回归分析。对于二维空间线性是一条直线；对于三维空间线性是一个平面，对于多维空间线性是一个超平面...

对于一元线性回归模型，假设从总体中获取了n组观察值（X1，Y1），（X2，Y2），...，（Xn，Yn）。对于平面中的这n个点，可以使用无数条曲线来拟合。要求样本回归函数尽可能好地拟合这组值。综合起来看，这条直线处于样本数据的中心位置最合理。 选择最佳拟合曲线的标准可以确定为：使总的拟合误差（即总残差）达到最小。有以下三个标准可以选择：

（1）用"残差和最小"确定直线位置是一个途径。但很快发现计算"残差和"存在相互抵消的问题。
（2）用"残差绝对值和最小"确定直线位置也是一个途径。但绝对值的计算比较麻烦。
（3）最小二乘法的原则是以"残差平方和最小"确定直线位置。用最小二乘法除了计算比较方便外，得到的估计量还具有优良特性。这种方法对异常值非常敏感。

最常用的是普通最小二乘法（ Ordinary Least Square，OLS）：所选择的回归模型应该使所有观察值的残差平方和达到最小。（Q为残差平方和）- 即采用平方损失函数。

样本回归模型：

$$Y_i = \hat{\beta}_0 + \hat{\beta}_1 X_i + e_i$$
$$\Rightarrow e_i = Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_i$$

其中e_i为样本（X_i, Y_i）的误差

平方损失函数：

$$Q = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 = \sum_{i=1}^{n} (Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_i)^2$$

则通过Q最小确定这条直线，即确定$\hat{\beta}_0, \hat{\beta}_1$，以$\hat{\beta}_0, \hat{\beta}_1$为变量，把它们看作是Q的函数，就变成了一个求极值的问题，可以通过求导数得到。求Q对两个待估参数的偏导数：

$$\begin{cases} \dfrac{\partial Q}{\partial \hat{\beta}_0} = 2\sum_{i=1}^{n} (Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_i)(-1) = 0 \\ \dfrac{\partial Q}{\partial \hat{\beta}_1} = 2\sum_{i=1}^{n} (Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_i)(-X_i) = 0 \end{cases}$$

根据数学知识我们知道，函数的极值点为偏导为0的点。

解得：

$$\hat{\beta}_2 = \frac{n\sum X_i Y_i - \sum X_i \sum Y_i}{n\sum X_i^2 - (\sum X_i)^2}$$

$$\hat{\beta}_1 = \frac{\sum X_i^2 \sum Y_i - \sum X_i \sum X_i Y_i}{n\sum X_i^2 - (\sum X_i)^2}$$

这就是最小二乘法的解法，就是求得平方损失函数的极值点。

< 2013年5月 >

日	一	二	三	四	五	六
28	29	30	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

导航
博客园
首页
新随笔
联系
订阅XML
管理

统计
随笔 - 14
文章 - 0
评论 - 1
引用 - 0

公告
昵称：iamccme
园龄：3年11个月
粉丝：14
关注：0
+加关注

搜索
找找看
谷歌搜索

常用链接
我的随笔
我的评论
我的参与
最新评论
我的标签

随笔分类
C++(2)
机器学习(3)
面试题(7)
推荐系统(2)

随笔档案
2013年10月 (1)
2013年9月 (2)
2013年6月 (1)
2013年5月 (10)

最新评论
1. Re:关于序列的面试题2----------最大连续子序列和以及积
第三题 求数组绝对值和最小的连续子序列
感觉有点问题，道理好像是通的，可是结果不对，不符合实际情况，还没想明白为什么。。。
--wj704

阅读排行榜
1. 机器学习经典算法之-----最小二乘法(11467)
2. 《统计学习方法》 梯度下降的两种应用场景(1916)
3. 百度NLP面试题(696)
4. 关于贝叶斯分类中的二项独立模型和多项式模型(569)
5. 人人数据挖掘实习生面经(412)

评论排行榜
1. 关于序列的面试题2----------最大连续子序列和以及积(1)

推荐排行榜
1. 机器学习经典算法之-----最小二乘法(11)
2. 人人数据挖掘实习生面经(1)
3. 《统计学习方法》 梯度下降的两种应用场景(1)

三. C++实现代码

```
/*
最小二乘法C++实现
参数1为输入文件
输入 ： x
输出： 预测的y
*/
#include<iostream>
#include<fstream>
#include<vector>
using namespace std;

class LeastSquare{
    double a, b;
public:
    LeastSquare(const vector<double>& x, const vector<double>& y)
    {
        double t1=0, t2=0, t3=0, t4=0;
        for(int i=0; i<x.size(); ++i)
        {
            t1 += x[i]*x[i];
            t2 += x[i];
            t3 += x[i]*y[i];
            t4 += y[i];
        }
        a = (t3*x.size() - t2*t4) / (t1*x.size() - t2*t2);  // 求得β1
        b = (t1*t4 - t2*t3) / (t1*x.size() - t2*t2);        // 求得β2
    }

    double getY(const double x) const
    {
        return a*x + b;
    }

    void print() const
    {
        cout<<"y = "<<a<<"x + "<<b<<"\n";
    }

};

int main(int argc, char *argv[])
{
    if(argc != 2)
    {
        cout<<"Usage: DataFile.txt"<<endl;
        return -1;
    }
    else
    {
        vector<double> x;
        ifstream in(argv[1]);
        for(double d; in>>d; )
            x.push_back(d);
        int sz = x.size();
        vector<double> y(x.begin()+sz/2, x.end());
        x.resize(sz/2);

        LeastSquare ls(x, y);
        ls.print();

        cout<<"Input x:\n";
        double x0;
        while(cin>>x0)
        {
            cout<<"y = "<<ls.getY(x0)<<endl;
            cout<<"Input x:\n";
        }
    }
}
```

四. 最小二乘法与梯度下降法

最小二乘法跟梯度下降法都是通过求导来求损失函数的最小值，那它们有什么区别呢。

相同

1.本质相同：两种方法都是在给定已知数据（independent & dependent variables）的前提下对dependent variables算出出一个一般性的估值函数。然后对给定新数据的dependent variables进行估算。

2.目标相同：都是在已知数据的框架内，使得估算值与实际值的总平方差尽量更小（事实上未必一定要使用平方），估算值与实际值的总平方差的公式为：

$$\Delta = \frac{1}{2}\sum_{i=1}^{m}(f_\beta(\bar{x}_i) - y_i)^2$$

其中$\bar{x}_i$为第i组数据的independent variable，y_i为第i组数据的dependent variable，β为系数向量。

不同

1.实现方法和结果不同：最小二乘法是直接对Δ求导找出全局最小，是非迭代法。而梯度下降法是一种迭代法，先给定一个，然后向下降最快的方向调整，在若干次迭代之后找到局部最小。梯度下降法的缺点是到最小点的时候收敛速度变慢，并且对初始点的选择极为敏感，其改进大多是在这两方面下功夫。

参考： http://blog.csdn.net/qll125596718/article/details/8248249

分类: 机器学习

iamccme
关注 - 0
粉丝 - 14
+加关注

11 0

« 上一篇：剑指offer--面试题1
» 下一篇：协同过滤算法之—slope one算法

posted on 2013-05-15 21:12 iamccme 阅读(11466) 评论(0) 编辑 收藏