Ai\_6 实现梯度下降

梯度下降：其实就是求代价函数的最小值，但只是局部最小值，就是从出发点一直求导，导数为正向左走倒数为负数向右走。

实现思路：将数据导入，然后设置一个初始的函数，将x带入然后和实际值y相减，求出误差，根据公式求导，不需要求平方只需要将价值函数求导后的函数带入就行，最后再套入梯度下降公式就可以更新两个值了。

实现语言：Java

代码:

package hello;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

import java.util.List;

import java.util.Vector;

public class GradientDescent {

public static Double[] getTheta(List<Double[]> X, Double[] y) {

//初始化长度

int m = y.length;

//初始化theta

Double[] theta = new Double[X.size()];

double a = 0.001;

for (int i = 0; i < theta.length; i++) {

theta[i] = 0.0;

}

//迭代150000次

for (int i = 0; i < 150000; i++) {

//初始化temp,做替换用

Double[] temp = new Double[theta.length];

for (int j = 0; j < temp.length; j++) {

temp[j] = 0.0;

}

for (int j = 0; j < m; j++) {

Double sum = 0.0;

for (int k = 0; k < theta.length; k++) {

//在二元图形中,这里相当于k\*x+b\*1,三元相当于a\*x+b\*y+c\*1,以此类推

sum += theta[k] \* X.get(k)[j];

}

sum = sum - y[j];

for (int k = 0; k < theta.length; k++) {

temp[k] += sum \* X.get(k)[j];

}

}

for (int j = 0; j < theta.length; j++) {

//一起替换 同时更新

theta[j] -= a / m \* temp[j];

}

}

return theta;

}

public static void main(String[] args) throws IOException {

Double[] x1 = GradientDescent.read("C:/Users/18242/Desktop/x.txt");

Double[] y = GradientDescent.read("C:/Users/18242/Desktop/y.txt");

int m = y.length;

Double[] x0 = new Double[m];

for (int i = 0; i < x0.length; i++) {

x0[i] = 1.0;

}

List<Double[]> X = new Vector<Double[]>();

X.add(x0);

X.add(x1);

Double[] theta = GradientDescent.getTheta(X, y);

for (int i = 0; i < theta.length; i++) {

System.out.println(theta[i]);

}

}

public static Double[] read(String fileName) throws IOException {

File file = new File(fileName);

FileReader fileReader = new FileReader(file);

BufferedReader reader = new BufferedReader(fileReader);

StringBuilder sb = new StringBuilder();

String str = reader.readLine();

while (str != null) {

sb.append(str);

str = reader.readLine();

}

reader.close();

fileReader.close();

String[] X0 = sb.toString().replace(" ", "").split(",");

Double[] x0 = new Double[X0.length];

for (int i = 0; i < x0.length; i++) {

x0[i] = Double.parseDouble(X0[i]);

}

return x0;

}

}

实现结果: