بسمه تعالی

عنوان درس : مباحث پیشرفته در زبانهای برنامه نویسی موازی

استاد : آقای دکتر سوادی

پروژه : پیاده سازی و موازی سازی الگوریتم Gradient descend linear regression

اعضاء گروه : محمد دانش آموز – مهسا زاهدی

توضیحات مربوط به این الگوریتم :

در این الگوریتم یکسری داده های آموزشی که در قالب یک ماتریس که سطر های آن داده های آموزشی و ستونهای آن مقادیر متناظر با پارامترهای هر داده آموزشی میباشد، به آن داده شده و نهایتا الگوریتم وزن ها (ضرایب پارامتر ها) را محاسبه می نماید.

این الگوریتم با انتخاب یکسری ضرایب تصادفی آغاز میشود سپس در هر مرحله با ضرب این ضرایب در داده های آموزشی و محاسبه خطای آموزشی الگوریتم ادامه می یابد. برای جلوگیری از تقابل خطاهای مثبت و منفی و همچنین بدست آوردن یک تابع خطا با قابلیت مشتق گیری پیوسته ، از مجموع مربعات خطای نمونه های آموزشی استفاده کرده و سعی در مینیمم کردن این تابع میگردد تا نهایتا با همگرا شدن این خطاها در هر اجرای الگوریتم، مقدار ضرایب با تولید کمترین میزان خطا بدست آید.

روش کار الگوریتم دقیقا براساس تغییر مقادیر ضرایب در جهت خلاف مشتق تابع مربعات خطاست و به همین علت نام آن را گر ادیان کاهشی، گذارده اند. این کاهش با یک گام حرکت پیش میرود که به آن ابر پارامتر آلفا میگوییم. با انتخاب مقادیر خیلی کوچک این گام حرکت، الگوریتم در زمانی بسیار طولانی ممکن است همگرا شود و با انتخاب گام بزرگ، ممکن است حتی واگرا شود. به همین دلیل در پیاده سازی خود، گام حرکت را با مقدار نسبتا بزرگ انتخاب میکنیم و در صورت واگرا شدن در هر مرحله آ آنرا با تقسیم بر یک ثابت بزرگتر از یک، کوچک میکنیم تا الگوریتم دوباره به روال همگرا شدن باز گردد.

از آنجایی که در اغلب بخشهای این الگوریتم از ضرب ماتریس و بردار استفاده میشود، بصورت بالقوه قابلیت خوبی برای موازی سازی دارد.

پیاده سازی اولیه این الگوریتم بصورت سریال در لینک و تصویر زیر ، قابل مشاهده است.

https://github.com/mohammaddan/ferdowsi-gradient-descend-regression

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <cmath>
#include <algorithm>
using namespace std;

const int n=200;
const int p=10;
const double epsilon=0.0000000001;
double params[]={100,2,4,0.5,1,4,3,2,1,2};
int attrbite_range[]={10,100,50000,5,10,3,50,20,50,100};
double norm[p];
```

```
double train_x[n][p]={0},
        train_y[n]={0};
double theta[p];

void set_train_set();
void tink(double* theta);
double cal_y(double* x);
double predict_y(double* x);
void setcolor(int,int=0);
void print_vector(string name,double* t,int n,int foreground=7,int
background=0);

int main(int argc, const char** argv) {
    srand(time(NULL));
    set_train_set();
    tink(theta);

    print_vector("theta",theta,p,2,0);
    cout<<"\n";
    double x[p];
    x[0]=1;
    for(int i=1;i<p;i++) x[i]=attrbite_range[i]/(1+rand()%5);
    print_vector("x",x,p,7,4);
    double cy=cal_y(x),py=predict_y(x);
    setcolor(2,0);
    cout<<"  y="<<cy<<"  predict_y="<<py;
    setcolor(1);
    cout<<" diff="<<fabs(cy-py)<<"\n\n";
    setcolor(7);
    return 0;
}

double cal_y(double* x){
    double result=theta[0];
    for(int i=1;i<p;i++) result+=x[i]*theta[i];
    return result;
}

double predict_y(double* x){
    double result=params[0];
    for(int i=1;i<p;i++) result+=x[i]*params[i];
    return result;
}

void set_train_set(){
    double t=0;
    for(int i=0;i<n;i++){
        train_x[i][0]=1;
        train_y[i]=train_x[i][0]*params[0];
        for(int j=1;j<p;j++) {
            float noise=(rand()%100-50)/10000.0;
            train_x[i][j]=rand()%attrbite_range[j];
            train_x[i][j]/=attrbite_range[j];
            train_y[i]+=params[j]*train_x[i][j]+noise;
        }
    }
}
```

```
void tink(double* theta){
    double alpha=0.01;
    for(int i=0;i<p;i++) theta[i]=(rand()%10)/1000.0;
    double delta[p]={0};
    double y_pred[n]={0};
    int cnt=2000000,it=0;
    double last_error,error=100;
    do{
        last_error=error;
        for(int i=0;i<p;i++) delta[i]=0;
        for(int i=0;i<n;i++){
            y_pred[i]=0;
            for(int j=0;j<p;j++){
                y_pred[i]+=theta[j]*train_x[i][j];
            }
        }
        double error2=0;
        for(int j=0;j<p;j++){
            for(int i=0;i<n;i++){
                double err=y_pred[i]-train_y[i];
                error2+=err*err;
                delta[j]+=train_x[i][j]*err;
            }
        }
        error=sqrt(error2);
        if(error>last_error) alpha/=1.2;
        for(int i=0;i<p;i++) theta[i] -= alpha*delta[i]/n;
        if(it%10000==0){
            cout<<"iterate : "<<it<<"\n";
            print_vector("theta",theta,p,2,0);
            cout<<"\nalpha="<<alpha<<" error : "<<error<<endl;
        }

    }while(it++<cnt && fabs(error-last_error)>epsilon);
    cout<<"\ncalulate on ";
    setcolor(4);
    cout<<it;
    setcolor(15);
    cout<<" iteratation \n";
    cout<<endl;
}

void print_vector(string name,double* t,int n,int foreground,int background){
    setcolor(foreground, background);
    cout<<name<<"={ ";
    for(int i=0;i<n;i++){
        cout<<t[i];
        if(i<n-1) cout<<",";
    }
    cout<<" }";
    setcolor(15);
}

void setcolor(int foreground,int background){
    foreground<8?foreground+=30:foreground+=82;
    background<8?background+=40:background+=92;
```

```
    if(background==40){
        cout<<"\033["<<foreground<<";0m";
        cout<<"\033["<<foreground<<"m";
    }
    else
        cout<<"\033["<<foreground<<";"<<background<<"m";
}
```

نمونه ای از اجرای برنامه