高级编程技术实验报告

实验五：模拟全球变暖

数据科学与计算机学院　17大数据与人工智能

17341015　陈鸿峥

一、 问题描述、求解思路及实验结果

Part A：创建模型

(i) 曲线拟合

`generate_models`：直接将`pylab.polyfit(x,y,deg)`的结果添加入结果列表中即可

(ii) 计算R^2

依照公式

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - e_i)^2}{\sum_{i=1}^{n}(y_i - \text{mean})^2}$$

对应进行计算即可，注意这里用到了类numpy的向量运算

```
def r_squared(y, estimated):
    mean = sum(y) / len(y)
    numerator = sum((y - estimated)**2)
    denominator = sum((y - mean)**2)
    return 1 - numerator / denominator
```

(iii) 模型可视化

`evaluate_models_on_training`的实施过程如下

- 枚举`models`中的每一个模型
- 创建估计值数组：`estimated = pylab.zeros(len(x))`
- 利用Python数组的负数索引，还原出拟合模型的多项式表达式，通过不断累加得到估计值，注意这里同样是进行向量运算

```
for i in range(len(model)):
    estimated += model[-i-1] * x**i
```

- 计算`r_squared`和`se_over_slope`
- 利用`plot`函数画图，同时利用`xlabel`、`ylabel`和`title`对图像进行标识

(iv) 调查趋势

依照题目要求，先从`climate`中获取必要的日/年份信息，然后用`generate_models`生成模型，再用`evaluate_models_on_training`进行模型评估。

注意对于年温度来说，需要对该年所有气温求均值，该年是否为闰年需要小心判断。代码细节如下。

```
climate = Climate("data.csv")
years = pylab.array(TRAINING_INTERVAL)

# Part A.4
# I. January 10th
temperature = [climate.get_daily_temp("NEW YORK",1,10,year) for year in years]
temperature = pylab.array(temperature)
model = generate_models(years,temperature,[1])
evaluate_models_on_training(years,temperature,model)

# II. Annual Temperature
temperature = []
for year in years:
    yearly_temp = pylab.sum(climate.get_yearly_temp("NEW YORK",year))
    temperature.append(yearly_temp / (366 if (year % 4 == 0 and year % 100 != 0)
        ↪ or (year % 400 == 0) else 365))
temperature = pylab.array(temperature)
model = generate_models(years,temperature,[1])
evaluate_models_on_training(years,temperature,model)
```

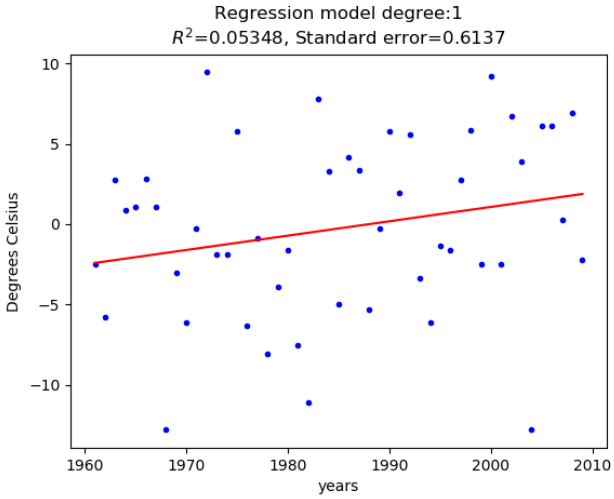

图 1: 问题4.I结果

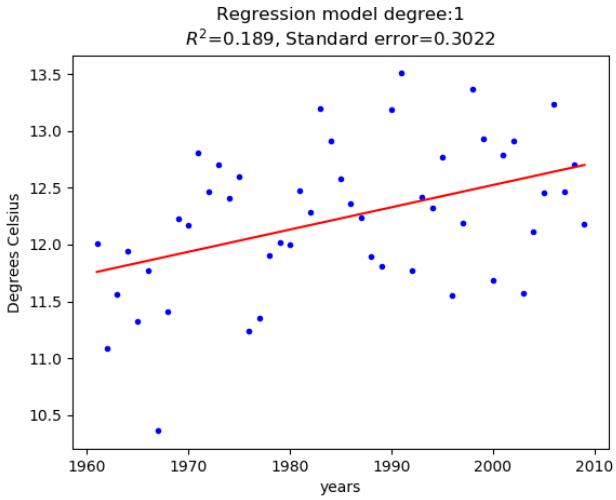

图 2: 问题4.II结果

Writeup回答

- 日气温的R^2更小，但年气温的拟合效果更好（这可能是因为噪声少导致的，见下题）。不管是日气温还是年气温R^2都远小于0.5，故气温上升趋势的显著性非常大。
- 因为每日/每年的气温具有随机性，因此噪声非常多。日气温的噪声更严重，因为日与日之间的差异很可能很大，而一整年的气温平均下来可以一定程度上消除噪声。
- 问题一已经回答，从图中确实可以看出气候在变暖。红色上升的拟合线及非常小的R^2都证实了气温上升，且可信度很高，标准误差都在0.5摄氏度左右。

Part B：处理更多数据

`gen_cities_avg`要注意遍历的方式及求平均的对象。本题中应该对某一城市某一特定年份的所有天数气温求平均后，再对该年所有城市的平均气温求平均，如下代码所示。

```
def gen_cities_avg(climate, multi_cities, years):
    res = []
    for year in years:
        temperature = []
        for city in multi_cities:
            yearly_temp = pylab.sum(climate.get_yearly_temp(city,year))
            temperature.append(yearly_temp / (366 if (year % 4 == 0 and year % 100
                ↪ != 0) or (year % 400 == 0) else 365))
        res.append(sum(temperature) / len(multi_cities))
    return pylab.array(res)
```

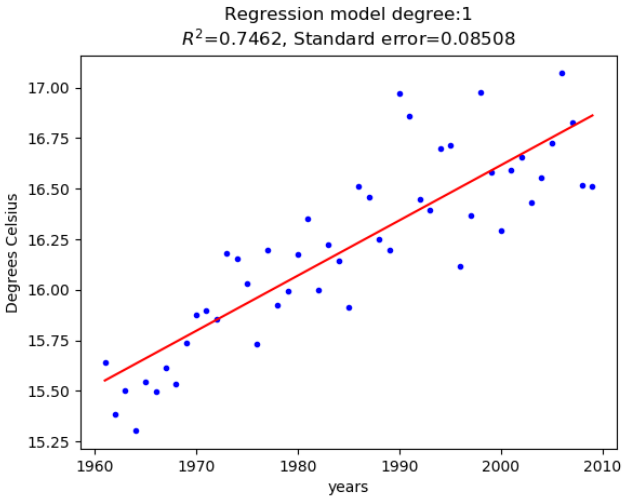

图 3: 问题B结果

Writeup回答

- R^2变大了，拟合出来的直线依然呈上升趋势，且标准误差非常小，故同样能说明气候在变暖
- 因为拟合出来的直线呈上升趋势，且标准误差非常小
- 如果只用3个不同城市，随机性较大，故噪声很多，气温上升变化没有那么明显；如果使用100个不同城市，由于样本量足够，故噪声相对较小，应该能表现出明显的气温上升
- 噪声会变得更小（拟合误差变小），因为相近区域的气温差距一般不会太大

Part C：移动平均

`moving_average`主要问题在于求和范围的选择，特别要注意左边界条件。利用Python的sliding技巧，可以很快写出以下代码。

```
def moving_average(y, window_length):
    res = []
    for i in range(len(y)):
        if i < window_length:
            mean = pylab.mean(y[0:i+1])
        else:
            mean = pylab.mean(y[i-window_length+1:i+1])
        res.append(mean)
    return pylab.array(res)
```

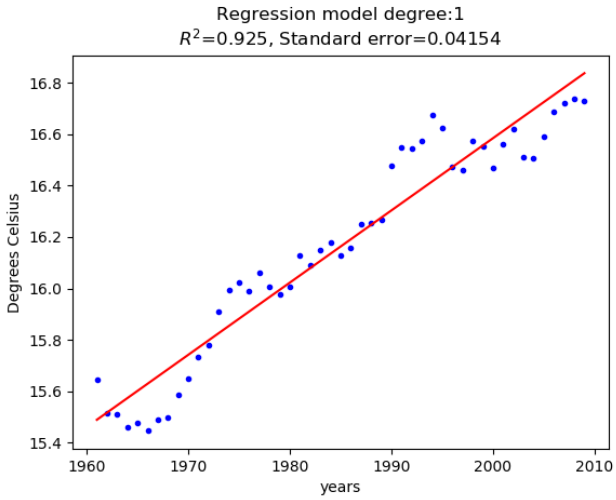

图 4: 问题C结果

Writeup回答

- 拟合得更加好，点基本均匀分布在直线两侧，R^2变得更大，误差变得更小。
- 因为把邻近5年的数据都取了平均，进一步消除随机性带来的噪声，从而拟合得更好，数据点与直线间的距离更小。

Part D：预测未来

(v) RMSE

直接依照公式实现即可

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - e_i)^2}{n}}$$

`evaluate_models_on_testing`与`evaluate_models_on_training`实现方式类似，仅仅是画图内容不同，因此这里不再赘述，详情请见代码。

(vi) 预测

如下两个步骤，先生成更多模型（次数为1、2、20），然后用这些模型进行预测，并取移动平均。

```
# Part D.2
# I. Generate more models
temperature = gen_cities_avg(climate,CITIES,years)
temperature_moving_avg = moving_average(temperature,5)
model = generate_models(years,temperature_moving_avg,[1,2,20])
evaluate_models_on_training(years,temperature_moving_avg,model)

# II. Predict the results
test_temperature = gen_cities_avg(climate,CITIES,test_years)
```

```
test_temperature_moving_avg = moving_average(test_temperature,5)
evaluate_models_on_testing(test_years,test_temperature_moving_avg,model)
```

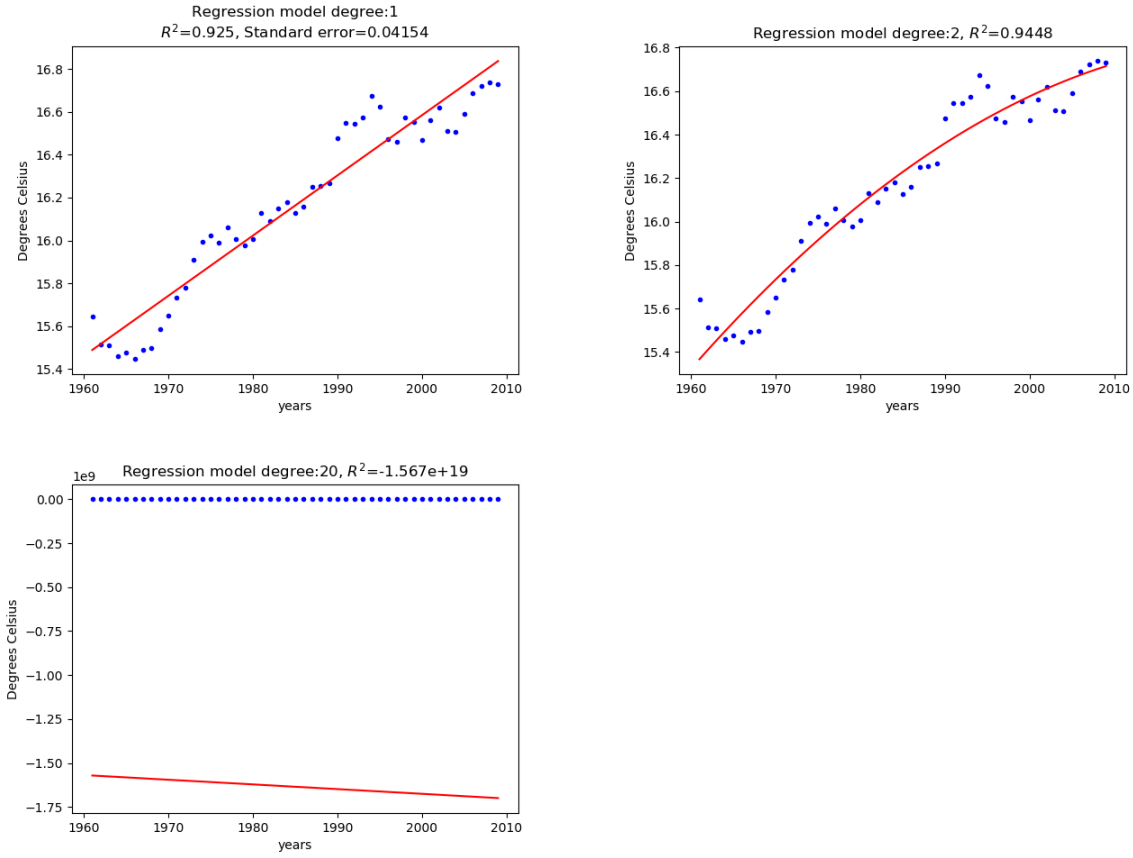

图 5: 问题D.I结果

Writeup I回答

- 度数为2的拟合效果最好，度数为20的拟合效果最差
- 度数为2的R^2最好，因为二次曲线能较好拟合数据，又不至于波动太大
- 度数为2的曲线拟合得最好，因为二次曲线度数刚刚好，能较好拟合数据，又不至于波动太大

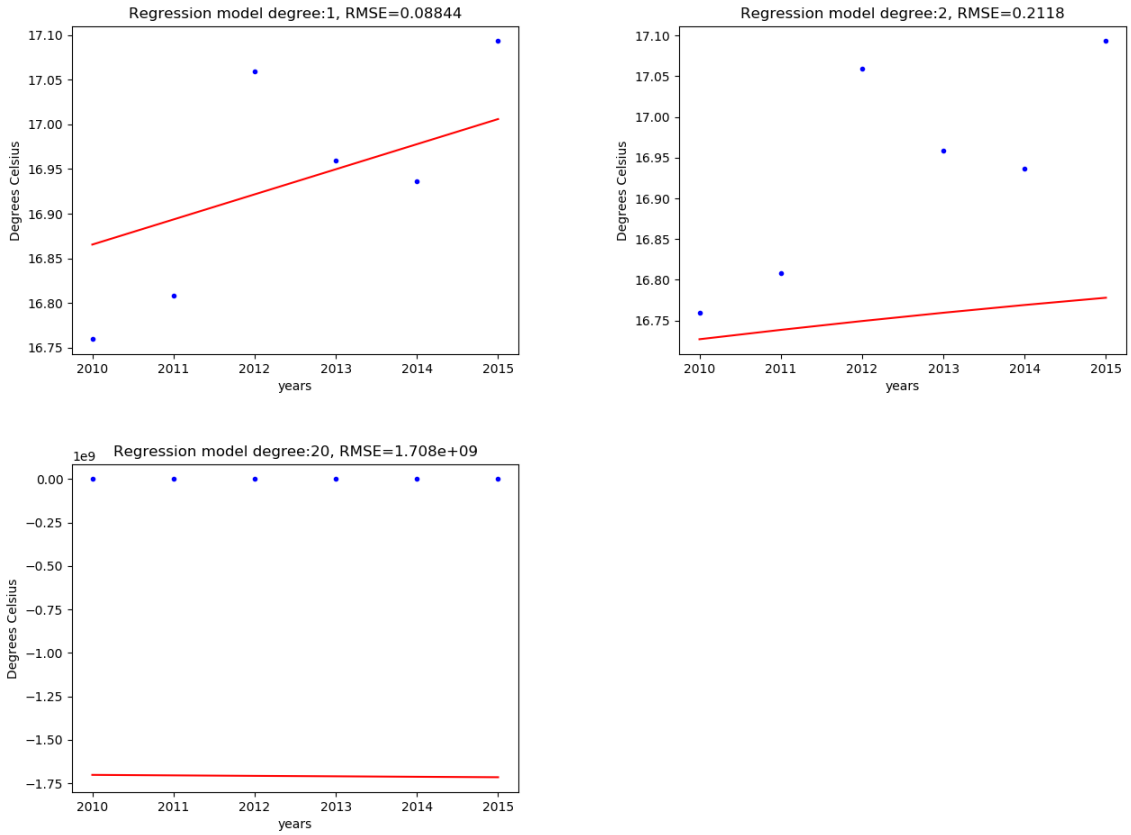

图 6: 问题D.II结果

Writeup II回答

- 度数为1的多项式预测效果比较合理，能使数据点均匀分布在两侧；度数为2和度数为20的多项式预测结果都偏低，所有实际数据点都在拟合曲线上方。关于RMSE，度数为1<度数为2<<度数为20。
- 线性预测得较好，度数为20的多项式最差。这与D.2.I的结果有点差异，二次曲线可能稍微有些过拟合，对未来的估计过为保守，故没有线性拟合的预测效果好。
- 见图7，预测结果将更加不稳定，但相比较之下，还是线性拟合的效果最好

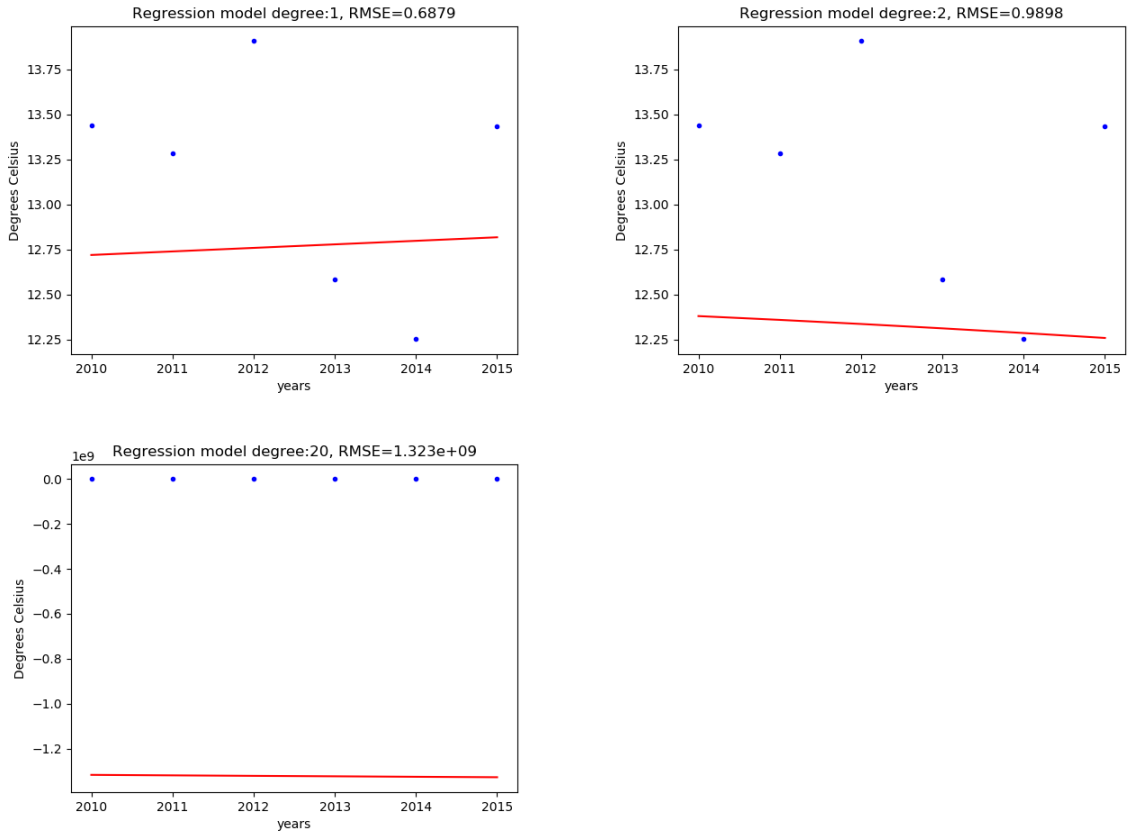

图 7: 问题D.II结果（用A.4.II数据拟合）

Part E：模拟极端气候

gen_std_devs与gen_cities_avg的实施类似，关键是mean的内容，这里应该对不同城市同一天的温度取平均，故axis=0。

```
def gen_std_devs(climate, multi_cities, years):
    res = []
    for year in years:
        temperature = []
        for city in multi_cities:
            temperature.append(climate.get_yearly_temp(city,year))
        daily_avg = pylab.mean(pylab.array(temperature),axis=0)
        res.append(pylab.std(daily_avg))
    return pylab.array(res)
```

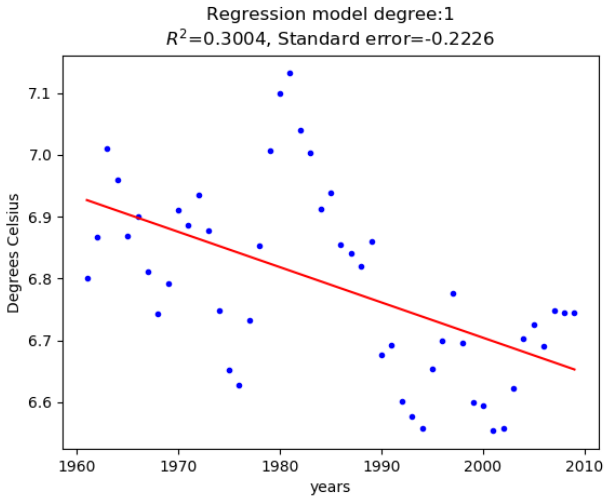

图 8: 问题E结果

Writeup回答

- 气温的变化在变得越来越小
- 收集更多城市的数据；采用拟合程度更加好的模型，如埃尔米特插值等

二、 代码

代码实施及注释请见附件`ps5.py`。

三、 其他实验结果

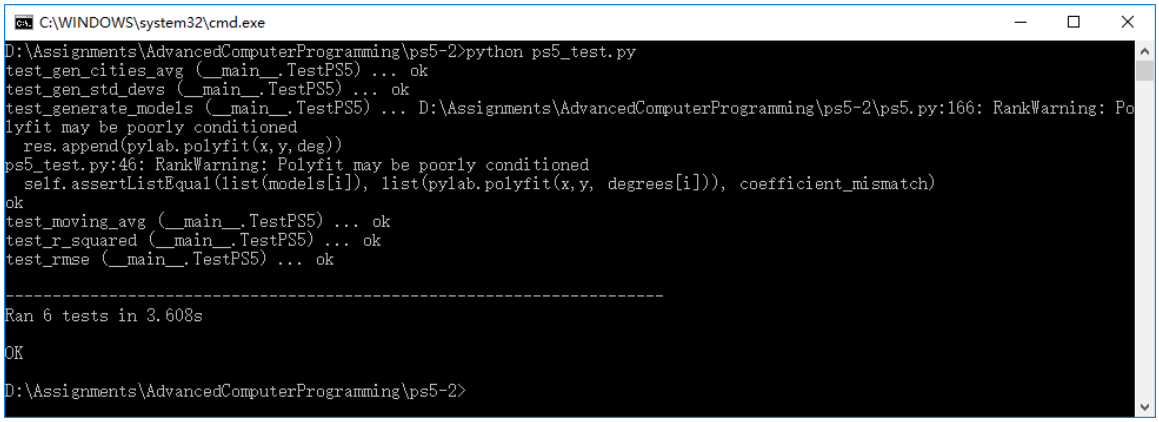

图 9: 运行ps5_tests.py的结果，可以看出所有测试样例通过