合肥工業大學

机器学习工程报告

课　　程： 机器学习基础

姓　　名： 王家琪

学　　号： 2018217918

完成时间： 2020 年 12 月 22 日

一． 工程摘要与每人贡献

摘要：

收集合肥地区过去一段时间(例如过去一年每个月的平均值)的空气质量(例如 pm2.5 值)，然后构建回归模型，能够预测今年某个月的空气质量值。

每人分工：

姓名	角色 (组长，组员，独自完成)	工作量 比例	负责内容
王家琪	独自完成	100%	收集数据集；算法代码实现；报告撰写

二、研究背景与意义

PM 是英文 particulate matter（颗粒物）的首字母缩写，PM2.5 指空气动力学当量直径小于或等于 2.5 微米（一微米等于百万分之一米）的悬浮颗粒物。它在大气中滞留时间长，传输距离远，含多种由于有害物质，而且与其他空气污染物存在着复杂的转化关系。PM2.5 易于滞留在终末细支气管和肺泡中，其中某些还可以穿透肺泡进入血液，也更易于吸附各种有毒的有机物和重金属元素，对健康的危害极大。

PM2.5 除来自自然界的风沙尘土、森林火灾、海水喷溅等，更主要来自工业生产、公路扬尘、建筑扬尘以及人类生产生活使用的能源燃烧等。

PM2.5 主要对呼吸系统和心血管系统造成伤害，包括呼吸道受刺激、咳嗽、呼吸困难、降低肺功能、加重哮喘、导致慢性支气管炎、心律失常、非致命性的心脏病、心肺病患者的过早死。老人、小孩以及心肺疾病患者是 PM2.5 污染的敏感人群。如果空气中 PM2.5 的浓度长期高于 10 微克/立方米，死亡风险就开始上升。浓度每增加 10 微克/立方米，总的死亡风险就上升 4%，得心肺疾病的死亡风险上升 6%，得肺癌的死亡风险上升 8%。

2012 年 10 月 11 日，中国国家环境保护部副部长吴晓青表示，新的《环境空气质量标准》颁布后，环保部明确提出了新标准实施的“三步走”目标。按照计划，2012 年年底前，京津冀、长三角、珠三角等重点区域以及直辖市、计划单列市和省会城市要按新标准开展监测并发布数据。截至目前，全国已有 195 个站点完成 PM2.5 仪器安装调试并试运行，有 138 个站点开始正式 PM2.5 监测并发布数据。

随着城市对 PM2.5 进行监测，人们在出行的时候更加的注意保护措施，环保意识也更加的强烈，自己的日常行为也在不断的改善。对空气中悬浮颗粒物、可吸入颗粒物等的关注程度得到了大大的提高，这在一定的程度上也促进国家对 PM2.5 监测和解决的进行。

三、模型方法

1.模型

采用普通的线性回归模型，使用每个数据帧样本中的 9 个 PM2.5 含量值：

$$y = \sum_{i=0}^{8} w_i x_i + b$$

x_i 为对应数据帧中第 i 个 PM2.5 含量，w_i 为其对应的权重值，b 为偏置项，y 为该数据帧样本的预测结果。

2.损失函数

用预测值与 label 之间的平均欧式距离来衡量预测的准确程度，并充当损失函数：

$$Loss_{\text{label}} = \frac{1}{2\text{num}} \sum_{n=0}^{num-1} (\hat{y}^n - y^n)^2$$

$\hat{y}^n$ 为第 n 个 label，y^n 为第 n 个数据帧的预测结果，num 为参加训练的数据帧样本个数。

加入正则项，防止过拟合，:

$$Loss_{regularization} = \frac{1}{2} \sum_{i=0}^{8} w_i^2$$

$$Loss = Loss_{\text{label}} + \beta \bullet Loss_{regularization} = \frac{1}{2}[\frac{1}{num} \sum_{n=0}^{num-1} (\hat{y}^n - y^n)^2 + \beta \sum_{i=0}^{8} w_i^2]$$

$Loss_{regularization}$ 为正则项，β 为正则项系数。

3.梯度更新

梯度计算：为使 Loss 最小，需要求 Loss 在 w 上的偏微分和 Loss 在 b 上的偏微分。

$$\frac{\partial Loss}{\partial w_i} = \frac{\partial Loss_{\text{label}}}{\partial y} \frac{\partial y}{\partial w_i} + \frac{\partial Loss_{regularization}}{\partial w_i} = \frac{1}{num} \sum_{n=0}^{num-1} (\hat{y}^n - \sum_{i=0}^{8} w_i x_i - b) \bullet (-x_i) + \beta \bullet \sum_{i=0}^{8} w_i$$

$$\frac{\partial Loss}{\partial b} = \frac{\partial Loss_{\text{label}}}{\partial y} \frac{\partial y}{\partial b} + \frac{\partial Loss_{regularization}}{\partial b} = \frac{1}{num} \sum_{n=0}^{num-1} (\hat{y}^n - \sum_{i=0}^{8} w_i x_i - b) \bullet (-1)$$

计算出梯度后，通过梯度下降法实现参数更新。

$$w_{newi} = w_i - \eta_w \frac{\partial Loss}{\partial w_i}, b_{new} = b - \eta_b \frac{\partial Loss}{\partial b}$$

η_w为权重 w 更新时的学习率，η_b为偏置 b 更新时的学习率。

4.学习率更新

采用 adagrad 算法来更新学习率。在不影响模型效果的前提下提高学习速度，对学习率进行实时更新，即让学习率的值在学习初期较大，之后逐渐减小。

$$\eta_n = \frac{\eta_{n-1}}{\sqrt{\sum_{i=1}^{n-1} grad_i^2}}$$

η_n为更新后的学习率，η_{n-1}为更新前的学习率，$\sqrt{\sum_{i=1}^{n-1} grad_i^2}$为在此之前所有梯度平方和的二次根。

四、系统设计

1.数据处理

2014/1/1	AMB_TEMI	14	14	14	13	12	12	12	12	15	17	20
2014/1/1	CH4	1.8	1.8	1.8	1.8	1.8	1.8	1.8	1.8	1.8	1.8	1.8
2014/1/1	CO	0.51	0.41	0.39	0.37	0.35	0.3	0.37	0.47	0.78	0.74	0.59
2014/1/1	NMHC	0.2	0.15	0.13	0.12	0.11	0.06	0.1	0.13	0.26	0.23	0.2
2014/1/1	NO	0.9	0.6	0.5	1.7	1.8	1.5	1.9	2.2	6.6	7.9	4.2
2014/1/1	NO2	16	9.2	8.2	6.9	6.8	3.8	6.9	7.8	15	21	14
2014/1/1	NOx	17	9.8	8.7	8.6	8.5	5.3	8.8	9.9	22	29	18
2014/1/1	O3	16	30	27	23	24	28	24	22	21	29	44
2014/1/1	PM10	56	50	48	35	25	12	4	2	11	38	56
2014/1/1	PM2.5	26	39	36	35	31	28	25	20	19	30	41
2014/1/1	RAINFALL	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR
2014/1/1	RH	77	68	67	74	72	73	74	73	66	56	45
2014/1/1	SO2	1.8	2	1.7	1.6	1.9	1.4	1.5	1.6	5.1	15	4.5
2014/1/1	THC	2	2	2	1.9	1.9	1.8	1.9	1.9	2.1	2	2
2014/1/1	WD_HR	37	80	57	76	110	106	101	104	124	46	241
2014/1/1	WIND_DIR	35	79	2.4	55	94	116	106	94	232	153	283
2014/1/1	WIND_SPE	1.4	1.8	1	0.6	1.7	2.5	2.5	2	0.6	0.8	1.6
2014/1/1	WS_HR	0.5	0.9	0.6	0.3	0.6	1.9	2	2	0.5	0.3	0.8

部分数据截图

数据中存在空数据 NR，RAINFALL 表示当天对应时间点是否降雨，有降雨值为 1，无降雨值为 NR，则将空数据 NR 替换为 0。

每一天包含的信息维度为(18,24)(18 项指标，24 个时间节点)。将 0 到 8 时的数据截取出来，形成一个维度为(18,9)的数据帧，作为训练数据，将 9 时的 PM2.5 含量取出来，作为该训练数据对应的 label；同理可取 1 到 9 时的数据作为训练用的数据帧，10 时的 PM2.5 含量作为 label。以此分割，可将每天的信息分割为 15 个 shape 为(18,9)的数据帧和 15 个 label。

训练集中共包含 240 天的数据，共可获得 3600 个数据帧和对应的 label。

```python
# 数据处理
def dataProcess(df):
    x_list, y_list = [], []
    df = df.replace(['NR'], [0.0])  # 将空数据NR替换为0
    array = np.array(df).astype(float)  # 转换数据类型
    for i in range(0, 4320, 18):    # 将数据集拆分为多个数据帧
        for j in range(24-9):
            mat = array[i:i+18, j:j+9]
            label = array[i+9, j+9]
            x_list.append(mat)
            y_list.append(label)
    x = np.array(x_list)
    y = np.array(y_list)
    return x, y, array
```

2. 训练模型

参照上文模型方法中所述：

```python
# 训练模型
def train(x_train, y_train, epoch):
    bias = 0    # 初始化偏置值
    weights = np.ones(9)    # 初始化权重
    learning_rate = 1   # 初始化学习率
    reg_rate = 0.001    # 正则项系数
    bg2_sum = 0     # 梯度平方和(存放偏置值)
    wg2_sum = np.zeros(9)   # 梯度平方和(存放权重)

    for i in range(epoch):
        b_g = 0
        w_g = np.zeros(9)
        for j in range(3200):   # 在所有数据上计算Loss_label的梯度
            b_g += (y_train[j] - weights.dot(x_train[j, 9, :]) - bias) * (-1)
            for k in range(9):
                w_g[k] += (y_train[j] - weights.dot(x_train[j, 9, :]) - bias) * (-x_train[j, 9, k])
        b_g /= 3200     # 求均值
        w_g /= 3200
        for m in range(9):  # 与Loss_regularization在w上的梯度相加
            w_g[m] += reg_rate * weights[m]
        bg2_sum += b_g**2   # adagrad算法
        wg2_sum += w_g**2
        bias -= learning_rate/bg2_sum**0.5 * b_g    # 更新权重和偏置
        weights -= learning_rate/wg2_sum**0.5 * w_g

    return weights, bias
```

3. 验证效果

在训练过程中，每训练 200 轮输出一次在训练集上的损失：

```
if i%200 == 0:  # 输出训练集上的损失 200轮/次
    loss = 0
    for j in range(3200):
        loss += (y_train[j] - weights.dot(x_train[j, 9, :]) - bias)**2
    print('{} 轮后, 训练集上的损失:'.format(i), loss/3200)
```

结束后，输出在验证集上的损失：

```
# 验证效果
def validate(x_val, y_val, weights, bias):
    loss = 0
    for i in range(400):
        loss += (y_val[i] - weights.dot(x_val[i, 9, :]) - bias)**2
    return loss / 400
```

五．实验结果分析、对比和讨论

1.实验结果

```
C:\Users\G1aft_w\AppData\Local\Programs\Python\Python38-32\python.exe D:/文档/机器学习/pmPredict.py
0 轮后, 训练集上的损失: 955.3009375
200 轮后, 训练集上的损失: 49.86823677027294
400 轮后, 训练集上的损失: 46.20101423801224
600 轮后, 训练集上的损失: 44.88913061600438
800 轮后, 训练集上的损失: 44.26903588227097
1000 轮后, 训练集上的损失: 43.95010919056685
1200 轮后, 训练集上的损失: 43.78092633274225
1400 轮后, 训练集上的损失: 43.68982565130423
1600 轮后, 训练集上的损失: 43.64031430329769
1800 轮后, 训练集上的损失: 43.613225892364426
验证集上的损失: 40.35422383809946

Process finished with exit code 0
```

模型在验证集上的损失为 40 左右，即预测值与 label 之间的平均差异在 6 到 7 之间，模型的整体效果较差。

2.改进思路

(1)更换模型，不使用线性回归模型，从数据中发现规律并以此为依据建立模型。

(2)数据集中除 PM2.5 的值外还有许多可能会影响或与 PM2.5 有关联的数据，在构建模型时，可充分考虑 PM2.5 与其他大气成分之间的关系，构建更合理的模型。

(3)分割训练集和验证集时，可以按照比例随机抽取数据帧作为训练集和验证集。

六．对本门课的感想、意见和建议

建议在本门课程前开设 python 语言相关课程，由于对 python 比较不熟悉，在进行实验的过程中遇到了很多问题并需要花费大量时间解决,在本门课程结束后我对此语言的使用更加熟练了。

个人感觉本门课程与之前所学专业课难度跨度较大，仅通过上课所学无法理解掌握，需要自己在课余时间查阅资料并进行实践。

随着大数据时代的发展和 AI 领域的延伸，我认为开设本门课程很有必要，可以让我们了解专业的前沿发展并培养相关领域的兴趣。