第一个机器学习样例

2017-04-20 | 绪论

（本文会用到的所有代码都在这里）

作为“绪论”的总结，我们来运用 Python 解决一个实际问题以对机器学习有具体的感受吧。由于该样例只是为了提供直观，我们就拿比较有名的一个小问题来进行阐述。俗话云：“麻雀虽小，五脏俱全”，我们完全可以通过这个样例来对机器学习的一般性步骤进行一个大致的认知

该问题来自 Coursera 上斯坦福大学机器学习课程（which is 我的入坑课程），其叙述如下：现有包含 47 个房子的面积和价格，需要建立一个模型对新的房价进行预测。稍微翻译一下问题，可以得知：

- 输入数据只有一维、亦即房子的面积
- 目标数据也只有一维、亦即房子的价格
- 我们需要做的、就是根据已知的房子的面积和价格的关系进行机器学习

下面我们就来一步步地进行操作

获取与处理数据

原始数据集的前 10 个样本如下表所示，这里房子面积和房子价格的单位可以随意定夺、因为它们不会对结果造成影响：

房子面积	房子价格	房子面积	房子价格

2104	399900	1600	329900
2400	369000	1416	232000
3000	539900	1985	299900
1534	314900	1427	198999
1380	212000	1494	242500

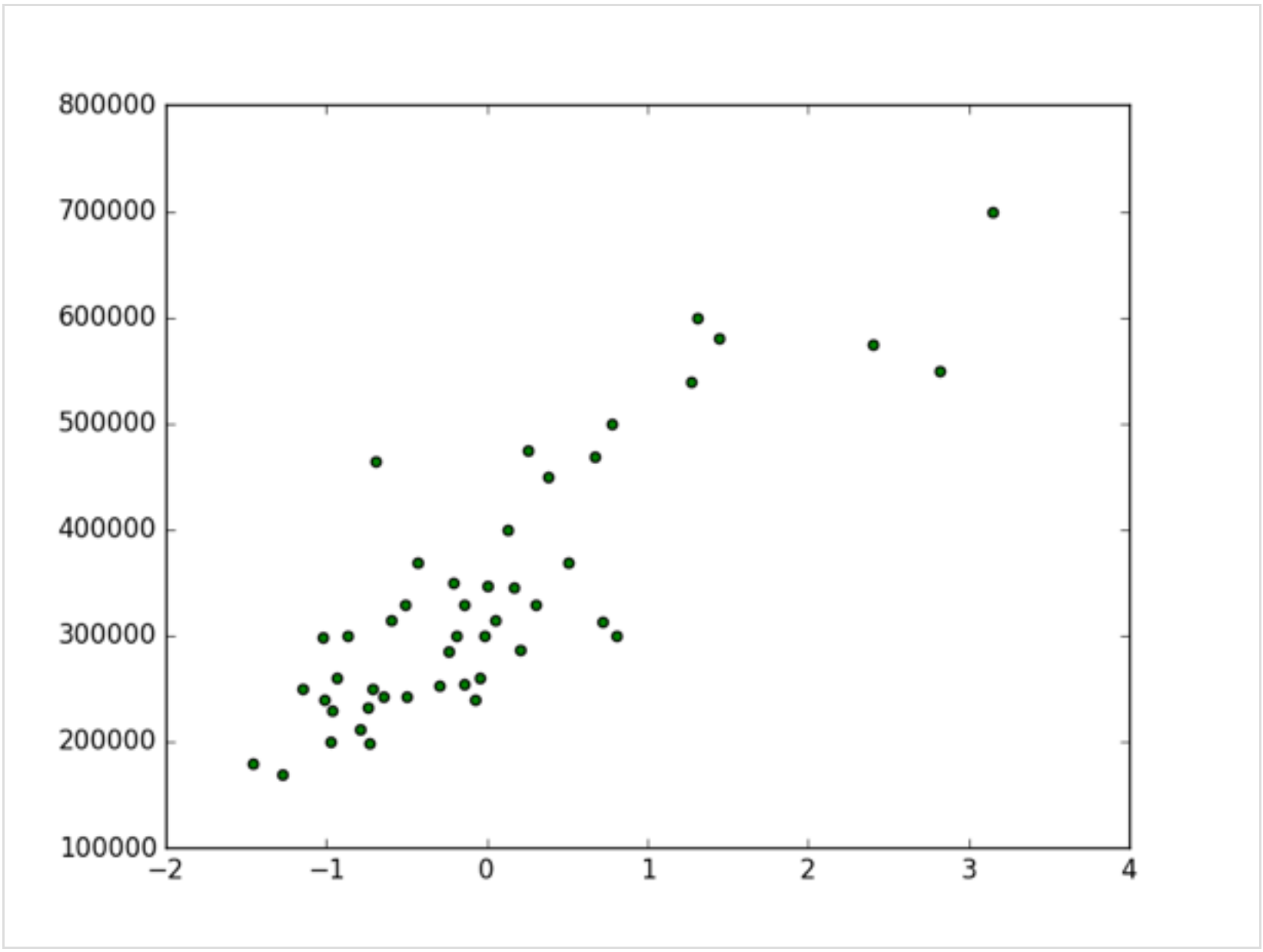
完整的数据集可以参见这里。虽然该数据集比较简单，但可以看到其中的数字都相当大。保留它原始形式确实有可能是有必要的，但一般而言、我们应该对它做简单的处理以期望能够降低问题的复杂度。在这个例子里，我们采取常用的、将输入数据标准化的做法，其数学公式为：

$$X = \frac{X - \bar{X}}{std(X)}$$

其中$\bar{X}$ $\bar{X}$表示X X（房子面积）的均值、$std(X)$ $std(X)$表示X X的标准差（Standard Deviation）。代码实现则如下：

```
# 导入需要用到的库
import numpy as np
import matplotlib.pyplot as plt

# 定义存储输入数据（x）和目标数据（y）的数组
x, y = [], []
# 遍历数据集，变量 sample 对应的正是一个个样本
for sample in open("../_Data/prices.txt", "r"):
    # 由于数据是用逗号隔开的，所以调用 Python 中的 split 方法并将逗号作为参数传入
    _x, _y = sample.split(",")
    # 将字符串数据转化为浮点数
    x.append(float(_x))
    y.append(float(_y))
# 读取完数据后，将它们转化为 Numpy 数组以方便进一步的处理
x, y = np.array(x), np.array(y)
# 标准化
x = (x - x.mean()) / x.std()
# 将原始数据以散点图的形式画出
plt.figure()
plt.scatter(x, y, c="g", s=20)
plt.show()
```

上面这段代码的运行结果如下图所示：



预处理后的数据散点图

预处理后的数据散点图

这里横轴是标准化后的房子面积，纵轴是房子价格。以上我们已经比较好地完成了机器学习任务的第一步：数据预处理

选择与训练模型

在弄好数据之后、下一步就要开始选择相应的学习方法和模型了。幸运的是，通过可视化原始数据，我们可以非常直观地感受到：我们很有可能通过线性回归（Linear Regression）中的多项式拟合来得到一个不错的结果。其模型的数学表达式如下：

注意：用多项式拟合散点只是线性回归的很小的一部分、但是它的直观意义比较明显。考虑到问题比较简单、我们才选用了多项式拟合。线性回归的详细讨论超出了本书的范围，这里不做赘述

$$f(x\,|\,p;n) = p_0 x^n + p_1 x^{n-1} + \ldots + p_{n-1} x + p_n L(p;n) = \frac{1}{2}\sum_{i=1}^{m}[f(x\,|\,p;n) - y]^2$$

其中$f(x|p;n)$$f(x\,|\,p;n)$就是我们的模型，`p`、`n`都是模型的参数，其中`p`是多项式`f`的各个系数、`n`是多

项式的次数。$L(p;n)L(p;n)$则是模型的损失函数，这里我们采用了常见的平方损失函数、也就是所谓的欧氏距离（或说向量的二范数）。`x`、`y`则分别是输入向量和目标向量；在我们这个样例中，`x`、`y`这两个向量都是 47 维的向量，分别由 47 个不同的房子面积、房子价格所构成

在确定好模型后，我们就可以开始编写代码来进行训练了。对于大多数机器学习算法，所谓的训练正是最小化某个损失函数的过程，我们这个多项式拟合的模型也不例外：我们的目的就是让上面定义的$L(p;n)$ $L(p;n)$最小。在数理统计领域里面有专门的理论研究这种回归问题，其中比较有名的正规方程更是直接给出了一个简单的解的通式。不过由于有 Numpy 的存在，这个训练过程甚至变得还要更加简单一些：

```
# 在(-2,4)这个区间上取 100 个点作为画图的基础
x0 = np.linspace(-2, 4, 100)

# 利用 Numpy 的函数定义训练并返回多项式回归模型的函数
# deg 参数代表着模型参数中的 n、亦即模型中多项式的次数
# 返回的模型能够根据输入的 x（默认是 x0）、返回相对应的预测的 y
def get_model(deg):
    return lambda input_x=x0: np.polyval(np.polyfit(x, y, deg), input_x)
```

这里需要解释 Numpy 里面带的两个函数：`polyfit`和`polyval`的用法：

- `polyfit(x, y, deg)`：该函数会返回使得上述（注：该公式中的`x`和`y`就是输入的`x`和`y`）$L(p;n)=\frac{1}{2}\sum_{i=1}^{m}[f(x|p;n)-y]^2L(p;n)=\frac{1}{2}\sum_{i=1}^{m}[f(x|p;n)-y]^2$最小的参数`p`、亦即多项式的各项系数。换句话说，该函数就是模型的训练函数
- `polyval(p, x)`：根据多项式的各项系数`p`和多项式中`x`的值、返回多项式的值`y`

评估与可视化结果

模型做好后、我们就要尝试判断各种参数下模型的好坏了。为简洁，我们采用$n=1,4,10n=1,4,10$这三组参数进行评估。由于我们训练的目的是最小化损失函数，所以用损失函数来衡量模型的好坏似乎是一个合理的做法：

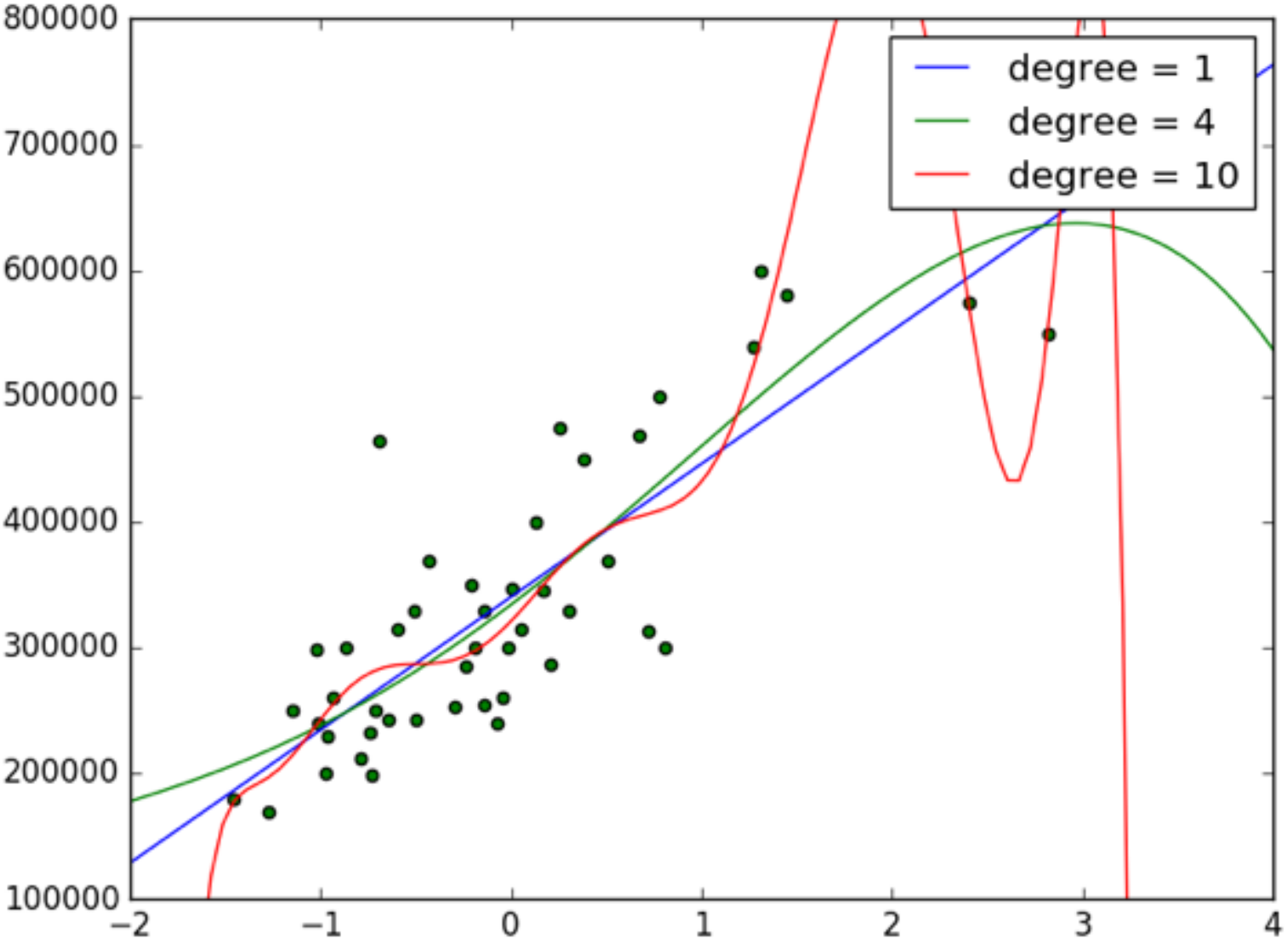
```
# 根据参数 n、输入的 x、y 返回相对应的损失
def get_cost(deg, input_x, input_y):
    return 0.5 * ((get_model(deg)(input_x) - input_y) ** 2).sum()

# 定义测试参数集并根据它进行各种实验
test_set = (1, 4, 10)
for d in test_set:
    # 输出相应的损失
    print(get_cost(d, x, y))
```

所得的结果是：当$n = 1, 4, 10$$n = 1, 4, 10$时，损失的头两个数字分别为 96、94 和 75。这么看来似乎是$n = 10$$n = 10$优于$n = 4$$n = 4$而$n = 1$$n = 1$最差；但从上面那张图可以看出，似乎直接选择$n = 1$$n = 1$作为模型的参数才是最好的选择。这里的矛盾的来源正是前文所提到过的过拟合情况

那么怎么最直观地了解是否出现过拟合了呢？当然还是画图了：

```
# 画出相应的图像
plt.scatter(x, y, c="g", s=20)
for d in test_set:
    plt.plot(x0, get_model(d)(), label="degree = {}".format(d))
plt.xlim(-2, 4)
# 将横轴、纵轴的范围分别限制在(-2,4)、(10^5,8 * 10^5)
plt.ylim(1e5, 8e5)
# 调用 legend 方法使曲线对应的 label 正确显示
plt.legend()
plt.show()
```

上面这段代码的运行结果如下图所示：



线性回归的可视化

线性回归的可视化

其中，蓝线、绿线、红线分别代表$n = 1$$n = 1$、$n = 4$$n = 4$、$n = 10$$n = 10$的情况（上图的右上角亦有说明）。可以看出，从$n = 4$$n = 4$开始模型就已经开始出现过拟合现象了，到$n = 10$$n = 10$时模型已经变得非常不合理

至此，可以说这个问题就已经基本解决了。在这个样例里面，除了交叉验证、我们涵盖了机器学习中的大部分主要步骤（之所以没有进行交叉验证是因为数据太少了……）。代码部分加起来总共 40~50 行，应该算是一个比较合适的长度。希望大家能够通过这个样例对机器学习有个大概的理解、也希望它能引起大家对机器学习的兴趣

观众老爷们能赏个脸么 (σ'ω')σ

赏

Python

人生苦短，我用 Python

“绪论”小结

撰写评论

发布

账号（邮件地址）

还没有评论，快来抢沙发吧！

正在载入来必力