Python编程及人工智能应用

第三章 线性回归及Python编程

https://bolei-zhang.github.io/course/python-ai.html

内容提要

- 线性回归问题简介
- 单变量线性回归问题
- 基于Scikit-learn库求解单变量线性回归
- 自定义求解单变量线性回归
  - 基于最小二乘法
  - 基于梯度下降法
- 多变量线性回归问题

人工智能 vs 机器学习

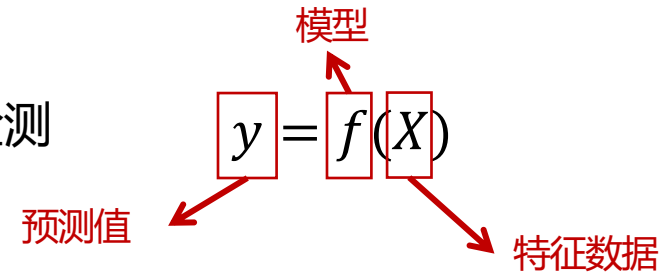
- 符号主义（Symbolism）：
  - 一种基于逻辑推理的智能模拟方法
  - 又称逻辑主义、心理学派或计算机学派。
  - 启发式算法→专家系统→知识工程，知识图谱

- 连接主义（Connectionism）：
  - 又称仿生学派或生理学派
  - 是一种基于神经网络及网络间的连接机制与学习算法的智能模拟方法，比如深度学习
  - 从历史经验中去学习

- 行为主义（Actionism）：
  - 强化学习：“感知——行动”的行为智能模拟方法

机器学习

- 机器学习
  - 让计算机模仿人类，从过去经验中学习一个“模型”，通过学到的模型再对新情况给出一个预测
  - “经验”通常是以“数据”的形式存在
- 机器学习“预测”场景
  - 根据房屋位置、面积、装修等预测房价
  - 根据图像预测图像的分类，对图像进行分割、检测
  - 机器翻译、自动问答、文本理解
- 监督学习
  - 分类（classification）：预测的值是“离散”的
  - 回归（regression）：预测的值是“连续”的



问题定义

- 数据集
  - 训练数据集(X_{train}, y_{train}) 特征 feature 标签 label

样本 instance

x_1	x_2	x_3	x_4	x_5	…	y
1.2	100	200	3	12		5.2
3.5	200	35	12	4.6		7.8
4.5	7.2	100	5	13.5		9.2
…	…	…	…	…		…

  - 测试数据集X_{test}

x_1	x_2	x_3	x_4	x_5	…	y
1.8	200	20	4	13		?
13.5	300	15	11	4.8		?
14.5	52	2	3	12.5		?

- 目标
  - 从训练数据集中学习模型f（如何形式化f）
  - 使得在测试数据集中$f(X_{test})$和真实的y_{test}尽量接近（如何定义接近）

线性回归问题与模型

- 基本形式
  - 给定有d个属性的样例$x = (x_1; x_2; x_3; \ldots; x_d)$，其中$x_i$是$x$在第$i$个属性上的取值，线性回归试图学习得到一个预测函数$f(x)$，该函数的值是各属性值的线性加权和，公式如下

$$f(x) = w_1 x_1 + w_2 x_2 + \ldots + w_d x_d + b$$

- 向量形式为：$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$
  - w和b是可学习和调整的参数，可自动从数据中学习获得
  - 预测某套商品房的总价

$$f(\mathbf{x}) = 3 \times \text{面积大小} + 0.5 \times \text{楼层指数} + 0.2 \times \text{卧室数量指数}$$

- 单变量线性回归与多变量线性回归

单变量线性回归问题

- 当样本仅1个属性时（即只有x_1），只要求解两个参数（w_1和b），是单变量线性回归模型

$$f(x) = w_1 x_1 + b$$

- 案例描述：设某小区通过某房产中介处已售出5套房，房屋总价与房屋面积之间有如下的数据关系。现有该小区的一位业主想要通过该房产中介出售房屋，在业主报出房屋面积后，根据训练数据，中介能否能估算出该房屋的合适挂售价格？

训练样本	房屋面积（平方米）	房屋总价（万元）
1	75	270
2	87	280
3	105	295
4	110	310
5	120	335

案例分析

•把房屋面积看成自变量x，房屋总价看成因变量y，先通过绘图看出二者之间的关系。

1. 房屋总价随着房屋面积的变化，大致呈现线性变化趋势；
2. 如果根据现有的训练样本数据能够拟合出一条直线，使之与各训练样本数据点都比较接近，那么根据该直线，就可以计算出任意房屋面积的房屋总价了。

```
    # 设置x轴和y轴的值域分别为70~130和240~350
    plt.axis([70, 130, 240, 350])
    plt.grid(True) #显示网格
    return plt
plt = initPlot()
xTrain = np.array([75, 87, 105,110,120])
yTrain = np.array([270,280,295,310,335])
plt.plot(xTrain, yTrain, 'k.') #k表示绘制颜色为黑色，点表示绘制散点图
plt.show()
```