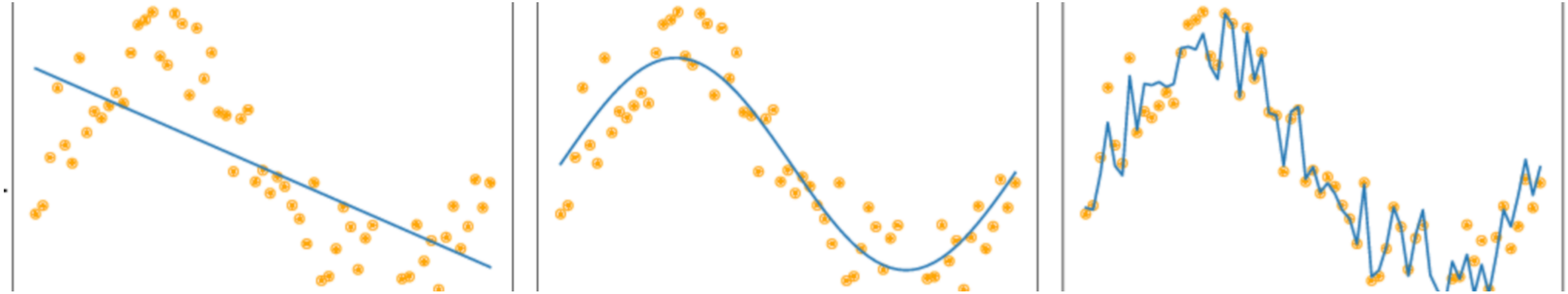
思考



以上三个模型，哪个拟合的最好？

本次课程将录像

Python编程及人工智能应用

第四章 逻辑斯蒂分类（Logistic Regression）及Python实现

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

Python编程及人工智能应用 支撑的毕业要求

- 目的：本课程是计算机专业的大学生所需要掌握的一门计算机专业课程。本课程的教学目的是，通过**理论教学**与**上机实践**，使学生**掌握Python语言，掌握人工智能基础算法，能使用Python语言实现人工智能相关算法，解决人工智能相关问题**。通过该课程，初步培养计算机专业学生人工智能相关领域的研究和应用能力。

- 知识单元七: 逻辑斯蒂分类及Python实现 （4学时）
  - **（1）知识点一：二分类逻辑斯蒂分类问题**
  - **（2）知识点二：基于Scikit-learn库的LogisticRegression类编码实现**
  - **（3）知识点三：基于梯度下降法编码实现**
  - （4）知识点四：分类模型的评价
  - （5）知识点五：非线性分类问题
  - （6）知识点六：正则化问题
  - （7）知识点七：多类别逻辑斯蒂分类
- 教学基本要求：
  - 掌握双类别逻辑分类基本原理，掌握调用Scikit-learn库函数进行双类别逻辑回归方法；掌握使用梯度下降法求解逻辑回归的方法，并能使用Python实现；掌握分类模型的评价方法；了解非线性分类问题和正则化问题；了解多类别逻辑斯蒂分类。

- 通过本课程的学习，使学生掌握使用Python进行人工智能算法编程，培养学生对人工智能学科的研究应用能力，形成对人工智能学科的兴趣。

内容提要

- 逻辑斯蒂分类简介
- 二分类逻辑斯蒂分类问题
- 基于Scikit-learn库求解二分类逻辑斯蒂分类
- 基于梯度下降法求解二分类逻辑斯蒂分类
- 分类模型的评价
- 非线性分类问题
- 正则化问题
- 多类别逻辑斯蒂分类问题

离散vs连续

鸭嘴兽体表有毛，**用乳汁哺乳后代**，具有哺乳动物的特征；但鸭嘴兽的**繁殖方式是卵生**，又像爬行动物。

- 离散化引入了不可觉察的误差来抵御外部的干扰。
- 其次，离散化简化了事务的描述方式，可以用简单的加减取代复杂的运算。
- 最后，离散化可以描述更多更复杂的用连续性无法描述的事务。

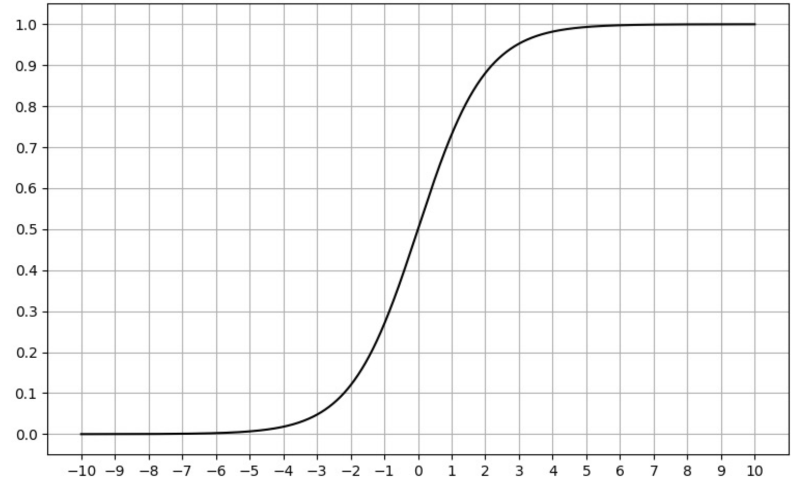
分类问题简介

- 分类问题的预测值是离散的
  - 根据晚风和晚霞预测明天是否晴天
  - 根据户型、面积、价格等因素预测房子是否好卖
  - 根据气色、打喷嚏、食欲等估算是否感冒
  - 根据西瓜的外观、敲瓜响声判断西瓜是否甜
  - 根据餐馆的飘香、入座情况等判断菜品是否好吃
- 分类对人类来说是一个基本能力
- 让人工智能学习分类是一个复杂的过程，需要优秀的模型、海量的数据和高性能的硬件支持

逻辑斯蒂分类简介

- 逻辑斯蒂分类（回归）（**Logistic** Regression），是一个回归方法，但通常用于**二分类**，也称为对数几率回归，逻辑回归
- 为了实现二分类，理想情况应该是一个单位阶跃函数
- 逻辑斯蒂分类通过拟合一个特殊的函数，即逻辑斯蒂函数（Logistic Function）进行分类

$$f(x)=\frac{1}{1+e^{-x}}$$



- f(x)值的取值范围在0~1之间
- 对于二分类问题，两个类分别用0和1表示
- 给定有d个属性的样例$x=(x_1;\ x_2;\ x_3;\ \ldots;\ x_d)$

$$P(y=1\mid\mathbf{x})=\frac{1}{1+e^{\mathbf{w}^T\mathbf{x}+b}} \qquad P(y=0\mid\mathbf{x})=\frac{e^{\mathbf{w}^T\mathbf{x}+b}}{1+e^{\mathbf{w}^T\mathbf{x}+b}}$$

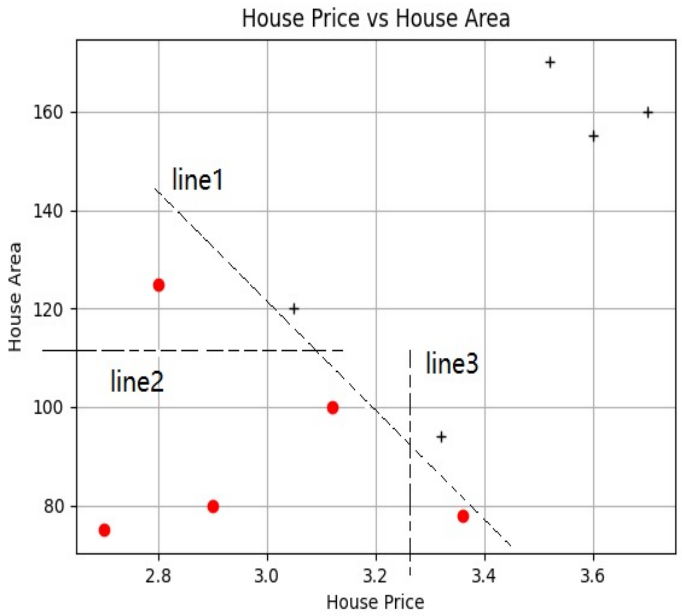
二分类逻辑斯蒂分类问题

- 当逻辑斯蒂分类类别数量只有两个时（即y的取值是0或1），是二分类逻辑斯蒂分类模型
- 案例描述：
  - 根据历史销售数据，该小区有些房屋好卖（在挂售半年内就可以成交），有些房屋不好卖（在挂售半年后还不能成交），观察发现，房屋是否好卖跟房屋挂售的房屋面积和每平方米的单价有很大关系。下表例举了15条历史销售记录，包括10条训练样本和5条测试样本。现有该小区的一位业主出售房屋，在业主报出房屋面积和期望售价后，根据表中的训练数据，**中介要判断该房屋是否好卖**。

样本	房屋面积	房屋单价（万元/平米）	是否好卖
训练样本1	78	3.36	是
训练样本2	75	2.70	是
训练样本3	80	2.90	是
训练样本4	100	3.12	是
训练样本5	125	2.80	是
训练样本6	94	3.32	否
训练样本7	120	3.05	否
训练样本8	160	3.70	否
训练样本9	170	3.52	否
训练样本10	155	3.60	否
测试样本1	100	3.00	是
测试样本2	93	3.25	否
测试样本3	163	3.63	是
测试样本4	120	2.82	是
测试样本5	89	3.37	是

案例分析

```
#代码4.1 表4.1中房屋销售数据的可视化展示代码
import numpy as np
import matplotlib.pyplot as plt
def initPlot():
    plt.figure()
    plt.title ( 'House Price vs House Area' )
    plt.xlabel ( 'House Price' ) #x轴标签文字
    plt.ylabel ( 'House Area' ) #y轴标签文字
    plt.grid ( True ) #显示网格
    return plt
xTrain0 = np.array ( [ [3.32, 94], [3.05, 120], [3.70, 160], [3.52, 170], [3.60, 155] ] ) #标注为不好卖的样本
yTrain0 = np.array ( [ 0, 0, 0, 0, 0 ] ) #y=0表示不好卖
xTrain1 = np.array ( [ [3.36, 78], [2.70, 75], [2.90, 80], [3.12, 100], [2.80, 125] ] ) #标注为好卖的样本
yTrain1 = np.array ( [ 1, 1, 1, 1, 1 ] ) #y=1表示好卖
plt = initPlot ()
plt.plot ( xTrain0 [ :, 0 ], xTrain0 [ :, 1 ], 'k+' ) #k表示黑色，+表示点的形状为十字
plt.plot ( xTrain1 [ :, 0 ], xTrain1 [ :, 1 ], 'ro' ) #r表示红色，o表示点的形状为圆形
plt.show ()
```



[1] https://matplotlib.org/stable/gallery/index

LogisiticRegression类

- 使用Scikit-learn库的LogisticRegression类解决逻辑斯蒂分类问题

from sklearn.linear_model import LogisiticRegression

model=LogisticRegression(penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='liblinear', max_iter=100, multi_class='ovr', verbose=0, warm_start=False, n_jobs=1)

  - penalty：正则化参数，可选值为 “L1” 和 “L2”
  - **solver：优化算法选择参数**
    - liblinear：使用坐标轴下降法来迭代优化损失函数
    - lbfgs：拟牛顿法的一种
    - newton-cg：也是牛顿法家族的一种
    - sag：随机平均梯度下降
  - multi_class：分类方式选择参数
  - class_weight：类别权重参数
  - **fit_intercept：是否存在截距**
  - **max_iter：算法收敛的最大迭代次数**
- 拟合函数fit（X, y）、预测函数predict（X）、预测概率函数predict_proba(X), 评价分数值score（X, y）

求解步骤

- 第一步：准备训练数据
  - xTrain = np.array ([[94], [120], [160], [170], [155], [78], [75], [80], [100], [125]])
  - yTrain = np.array ([0, 0, 0, 0, 0, 1, 1, 1, 1, 1])
- 第二步：创建LogisticRegression对象并拟合
  - from sklearn.linear_model import LogisticRegression #导入类
  - model = LogisticRegression (solver = "lbfgs") #创建对象，默认优化算法是L-BFGS
- 第三步：执行拟合
  - model.fit (xTrain, yTrain) #执行拟合
  - print (model.intercept_) #输出截距
  - print (model.coef_) #输出斜率
- 第四步：对新数据执行预测
  - newX = np.array ([[100], [130]]) #定义新样本
  - newY = print (model.predict (newX)) #输出斜率

编码实现

- 运行演示代码4.2

$$P(y=1|x)=f(x)=\frac{1}{1+e^{-(0.06426704x+7.23982418)}}$$

- 拟合得到函数：
  - 当x=112.65时，分母中的指数部分为零

$$P(y=1|x=112.65)=0.5$$

- 训练数据中有一个标记为“好卖”的样本（图中最右边的圆点）被分类函数错误地分类为“不好卖”（概率小于0.5，位于分割线的右边）
- 有一个标记为“不好卖”的样本（图中最左边的十字点）被分类函数错误地分类为“好卖”（概率大于0.5，位于分割线的左边）。

梯度下降法优化目标

- 逻辑斯蒂分类的判别函数 $P(y=1|\mathbf{x})=f(\mathbf{x})=\frac{e^{\mathbf{w}^T\mathbf{x}}}{1+e^{\mathbf{w}^T\mathbf{x}}}=\frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$
  - 其中： $\mathbf{w}^T=[w_0,w_1,w_2,...w_d]$ $\mathbf{x}^T=[1,x_1,x_2,...x_d]$
- 训练数据中有m个样本，$y^{(i)}$=0表示第i个样本的实际类别为第0类，$y^{(i)}$=1表示该样本的实际类别为第1类。
- M_0为实际类别为0的样本子集，M_1为实际类别为1的样本子集
  - 对于一个M_0中的样本i，其预测概率为$P(y=0|\mathbf{x}^{(i)})=1-\frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}^{(i)}}}$，**要尽量使得所有样本预测概率最大化**，这些样本满足独立同分布的性质；通常对这个函数取对数后进行优化

$$\max\frac{1}{|M_0|}\sum_{i\in M_0}\log(1-\frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}^{(i)}}})$$

  - 对于M_1中任一个样本i，其预测概率为$P(y=0|\mathbf{x}^{(i)})=\frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}^{(i)}}}$，**要尽量使得这个预测概率最大化**，这些样本满足独立同分布的性质；同样地，取对数后可得到

$$\max\frac{1}{|M_1|}\sum_{i\in M_0}\log(\frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}^{(i)}}})$$

梯度下降法优化目标

- 将以上两类样本的优化目标合并之后，可以得到总的优化目标如公式

$$\max imize \frac{1}{m}\sum_{i=1}^{m} y^{(i)}\log(\frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}^{(i)}}})+(1-y^{(i)})\log(1-\frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}^{(i)}}})$$

- 左半部分是用实际类别为1的训练样本进行优化，左半部分是用实际类别为0的训练样本进行优化
- 优化目标一般是进行最小化而不是最大化，L(w)也被称为损失函数（Loss Function）

$$L(\mathbf{w})=\frac{1}{m}\sum_{i=1}^{m} -y^{(i)}\log(\frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}^{(i)}}})-(1-y^{(i)})\log(1-\frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}^{(i)}}})$$

$$\min L(\mathbf{w})$$

梯度计算

- 梯度下降法需要根据梯度更新参数，更新公式如下

$$w_j = w_j - \alpha * \frac{\partial L(\mathbf{w})}{\partial w_j}$$

- 偏导数的求解如下（演示推导过程）

$$f(\mathbf{x}^{(i)}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}^{(i)}}}$$

$$\frac{\partial L(\mathbf{w})}{\partial w_j} = \frac{1}{m}\sum_{i=1}^{m} -y^{(i)}\frac{1}{f(\mathbf{x}(i))} \bullet \frac{\partial f(\mathbf{x}^{(i)})}{\partial \mathbf{w}_j} - (1-y^{(i)})\frac{1}{1-f(\mathbf{x}^{(i)})} \bullet \frac{-\partial f(\mathbf{x}^{(i)})}{\partial w_j}$$

$$\frac{\partial L(\mathbf{w})}{\partial w_j} = \frac{1}{m}\sum_{i=1}^{m}(-y^{(i)} + \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}^{(i)}}}) \bullet x_j^{(i)}$$

Python编码实现

- 演示运行代码4.3：基于梯度下降求解房价预测问题的Python实现

迭代次数： 176589

参数w0, w1, w2的值：

-1.9407385001273494 -3.8817781054764713 -4.408330368012581

输出结果的说明

- 该代码采用批量梯度下降法相同的实现，即bgd_optimizer函数。该函数需要传入成本函数（目标函数）、梯度函数、参数初始值、学习率等通用参数。具体到逻辑斯蒂分类问题，其成本函数和梯度函数已经在前面定义并用Python实现，作为参数传入bgd_optimizer函数即可。
- 由于“房屋面积”与“房屋单价”这两个属性具有不同取值范围，取值范围差异大，在样本数据传入梯度下降函数进行训练之前先进行归一化操作

$$x\_norm_i = \frac{x_i - \overline{x_i}}{std(x_i)}$$

- 训练数据的两个属性分别对应优化参数w1和w2，由于参数向量也包含w0，而w0与1对应，因此为了便于向量运算，在训练数据属性向量中增加一个值为1的量，对应代码中的make_ext()函数。
- 根据输出文字结果，梯度下降法总共迭代了176589次，得到的w0、w1、w2三个参数值约为-1.94、-3.88、-4.41，得到的逻辑斯蒂分类函数为

$$f(x) = \frac{1}{1+e^{-(-1.94-3.88x_1-4.41x_2)}}$$

回顾整个过程

1. 问题建模
   - 回归、分类 …

2. 收集数据
   - 回归：特征数据X，连续数据y
   - 分类：特征数据X，离散数据y

3. 特征预处理
   - 归一化
   - 类别特征
   - 时间特征
   - 图像数据、序列数据、图结构数据

4. 构建模型
   - 模型选择：线性回归、Logistic Regression
   - 损失函数：均方误差

5. 模型验证&参数调优
   - 使用训练数据训练模型，使用验证数据进行参数调优
   - 验证指标：回归（wmape、R2、均方误差）

6. 模型上线/AB测试
   - 在测试数据上进行模型测试（测试数据和训练数据来自于同一分布）

思考

- 假设有一类特征属于类别特征，比如房屋装修风格（中式、日式、简约、复古）
- 如何将这一类特征量化来输入模型？