

实验二：机器学习1-线性回归

第一关：数据载入与分析

一、实验目的

1、理解掌握线性回归模型原理，并利用工具库或包实现线性回归中编写数据载入，损失函数,梯度下降函数三部分。

2、学会使用 `sklearn` 构建线性回归算法，并利用经典数据集掌握线性回归分析过程。

二、实验内容

1、利用 `pandas`、`numpy` 构建一个完整的线性回归模型，主要编写数据载入，损失函数,梯度下降函数三部分。

2、使用 `sklearn` 构建线性回归算法，并利用已知标签的波士顿房价数据对模型进行训练，然后对未知标签的波士顿房价数据进行预测。

三、实验步骤

1、数据载入与分析。利用 `pandas` 读入数据 `data`，并将数据属性分别命名为 'Population'和'Profit'。

```
1. #encoding=utf8
2. import os
3. import pandas as pd
4.
5. if __name__ == "__main__":
6.     path = os.getcwd() + '/ex1data1.txt'
7.     #利用 pandas 读入数据 data，并将数据属性分别命名为'Population'和'Profit'
8.     #***** begin *****#
9.
10.    #***** end *****#
11.    print(data.shape)
```

2、计算损失函数:

$$h_{\theta}(x) = \theta^T x = \theta_0 + \theta_1 x_1$$

根据以上公式，编写计算损失函数`computeCost(X,y,theta)`，最后返回 `cost`。

X: 一元数据矩阵, 即 Population 数据;

y: 目标数据, 即 Profit 数据;

theta: 模型参数;

cost: 损失函数值。

```
1. #encoding=utf8
2. import numpy as np
3.
4. def computeCost(X, y, theta):
5.     #根据公式编写损失函数计算函数
6.     #***** begin *****#
7.
8.     #***** end *****#
9.     return cost
```

3、进行梯度下降得到线性模型:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)} - y^{(i)})x_j^{(i)})$$

根据以上公式, 编写计算损失函数 *gradientDescent(X,y,theta,alpha,itera)*,

最后返回theta, cost。

x: 一元数据矩阵, 即 Population 数据;

y: 目标数据, 即 Profit 数据;

theta: 模型参数;

m: 数据规模;

α : 学习率。

```
1. #encoding=utf8
2. import numpy as np
3.
4. def computeCost(X, y, theta):
5.     inner = np.power(((X * theta.T) - y), 2)
6.     return np.sum(inner) / (2 * len(X))
7.
8. def gradientDescent(X, y, theta, alpha, iters):
9.     temp = np.matrix(np.zeros(theta.shape))
10.    parameters = int(theta.ravel().shape[1])
11.    cost = np.zeros(iters)
```

```

12.
13.     for i in range(iters):
14.         error = (X * theta.T) - y
15.
16.         for j in range(parameters):
17.             ##### begin #####
18.
19.             ##### end #####
20.         theta = temp
21.         cost[i] = computeCost(X, y, theta)
22.
23.     return theta, cost

```

4、使用 `sklearn` 构建线性回归算法，并利用已知标签的波士顿房价数据对模型进行训练，然后对未知标签的波士顿房价数据进行预测。

波士顿房价数据集共有 506 条波士顿房价的数据，每条数据包括对指定房屋的 13 项数值型特征和目标房价组成。

在 `sklearn` 中通过 `LinearRegression` 方法实现线性回归。

`LinearRegression` 类中的 `fit` 函数用于训练模型，`fit` 函数有两个向量输入：

其中，`X` 大小为 [样本数量,特征数量] 的 `ndarray`，存放训练样本；`Y` 值为整型，大小为 [样本数量] 的 `ndarray`，存放训练样本的标签值。

`LinearRegression` 类中的 `predict` 函数用于预测，返回预测值，`predict` 函数有一个向量输入：

其中，`X` 大小为 [样本数量,特征数量] 的 `ndarray`，存放预测样本。

```

1.     #encoding=utf8
2.     from sklearn.linear_model import LinearRegression
3.     def lr(train_data,train_label,test_data):
4.         '''
5.         input:train_data 用来训练的数据
6.         train_label 用来训练的标签
7.         test_data 用来测试的数据
8.         '''
9.         ##### Begin #####
10.
11.         ##### End #####
12.     return predict

```

四、实验报告要求

参见实验报告模板

实验二、机器学习2-逻辑回归

一、实验目的

理解并掌握逻辑回归核心思想。

二、实验内容

- 1、根据理论知识编程实现 sigmoid 函数。
- 2、用 Python 构建梯度下降算法，并求取目标函数最小值。
- 3、使用 sklearn 构建逻辑回归算法，并利用癌细胞数据对模型进行训练，然后对未知的癌细胞数据进行识别。

三、实验步骤

- 1、根据所学知识完成编程题，实现 sigmoid 函数。

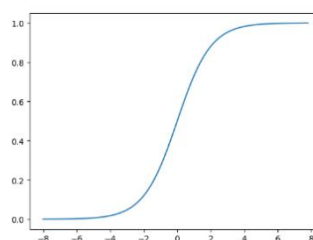
逻辑回归通过回归的思想来解决二分类问题的算法。逻辑回归将样本特征和样本所属类别的概率联系在一起，假设已经训练好了一个逻辑回归的模型为 $f(x)$ ，模型输出是样本 x 的标签是 1 的概率，则该模型可以表示 $\hat{p} = f(x)$ 。若得到了样本 x 属于标签 1 的概率后，能想到当 $\hat{p} > 0.5$ 时 x 属于标签 1，否则属于标签 0。所以有

$$\hat{y} = \begin{cases} 0 & \hat{p} < 0.5 \\ 1 & \hat{p} > 0.5 \end{cases}$$

逻辑回归中样本所属标签的概率和线性回归有关系，将线性回归的输出作为另一个函数的输入，该函数能够进行转换，假设函数为 σ ，转换后的概率为 $\hat{p}$ ，则逻辑回归在预测时可以看成 $\hat{p} = \sigma(W^T x + b)$ 。 σ 其实就是 sigmoid 函数：

$$\sigma(t) = 1 / (1 + e^{-t})$$

函数图像如下图所示：



从 sigmoid 函数图像可以看出当 t 趋近于 $-\infty$ 时函数值趋近于 0，当 t 趋

近于 $+\infty$ 时函数值趋近于 1。可见 sigmoid 函数的值域是 (0,1)，满足要将 $(-\infty, +\infty)$ 的实数转换成 (0,1) 的概率值的需求。因此逻辑回归在预测时可以看成：

$$\hat{p} = 1/(1 + e^{-W^T x + b})$$

```
1. #encoding=utf8
2. import numpy as np
3. #sigmoid 函数
4. def sigmoid(t):
5.     '''
6.     完成 sigmoid 函数计算
7.     :param t: 负无穷到正无穷的实数
8.     :return: 转换后的概率值
9.     '''
10.    #***** Begin *****#
11.
12.    #***** End *****#
```

2、用 Python 构建梯度下降算法，并求取目标函数最小值。

梯度下降的中心思想就是迭代地调整参数从而使损失函数最小化。通过测量参数向量 θ 相关的损失函数的局部梯度，并不断沿着降低梯度的方向调整，直到梯度降为 0，达到最小值。梯度下降公式如下：

$$\theta := \theta - \eta \nabla loss$$

对应到每个权重公式为：

$$w_i := w_i - \eta \frac{\partial loss}{\partial w_i}$$

其中 η 为学习率，是 0 到 1 之间的值，是个超参数，需要我们自己来确定大小。

梯度下降算法流程：

1. 随机初始参数；
2. 确定学习率；
3. 求出损失函数对参数梯度；
4. 按照公式更新参数；

重复 3、4 直到满足终止条件（如：损失函数或参数更新变化值小于某个阈值，或者训练次数达到设定阈值）。

```

1.  # -*- coding: utf-8 -*-
2.
3.  import numpy as np
4.  import warnings
5.  warnings.filterwarnings("ignore")
6.
7.  def gradient_descent(initial_theta,eta=0.05,n_iters=1000,epslion=1e-8):
8.      '''
9.      梯度下降
10.     :param initial_theta: 参数初始值, 类型为 float
11.     :param eta: 学习率, 类型为 float
12.     :param n_iters: 训练轮数, 类型为 int
13.     :param epslion: 容忍误差范围, 类型为 float
14.     :return: 训练后得到的参数
15.     '''
16.     # 请在此添加实现代码 #
17.     #***** Begin *****#
18.
19.     #***** End *****#

```

3、使用 `sklearn` 构建逻辑回归算法，并利用癌细胞数据对模型进行训练，然后对未知的癌细胞数据进行识别。

乳腺癌数据集，其实例数量是 569，实例中包括诊断类和属性，帮助预测的属性一共 30 个，各属性包括为 `radius` 半径（从中心到边缘上点的距离的平均值），`texture` 纹理（灰度值的标准偏差）等等，类包括：`WDBC-Malignant` 恶性和 `WDBC-Benign` 良性。用数据集的 80% 作为训练集，数据集的 20% 作为测试集，训练集和测试集中都包括特征和诊断类。

在 `sklearn` 中，使用 `LogisticRegression` 方法实现逻辑回归算法，`LogisticRegression` 的构造函数中有三个常用的参数可以设置：

- `solver`: { 'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga' }，分别为几种优化算法。默认为 `liblinear`。
- `C`：正则化系数的倒数，默认为 1.0，越小代表正则化越强。
- `max_iter`：最大训练轮数，默认为 100。和 `sklearn` 中其他分类器一样，`LogisticRegression` 类中的 `fit` 函数用于训练模型，`fit` 函数有两个向量输入：
- `X`：大小为 [样本数量,特征数量] 的 `ndarray`，存放训练样本。

- Y : 值为整型, 大小为 [样本数量] 的 ndarray, 存放训练样本的分类标签。LogisticRegression 类中的 predict 函数用于预测, 返回预测标签, predict 函数有一个向量输入:
- X : 大小为[样本数量,特征数量]的 ndarray, 存放预测样本。

LogisticRegression 的使用代码如下:

```
1. logreg = LogisticRegression(C=100)
2. logreg.fit(train_data, train_label)
3. predict = logreg.predict(test_data)
```

其中, train_data 为训练数据, train_label 为训练标签, test_data 为测试数据。predict 为癌细胞诊断结果。

在 begin-end 间补充代码, 使用 sklearn 实现逻辑回归算法。测试说明: 程序会调用你实现的方法对癌细胞数据进行识别, 正确率大于 0.95 则视为通关。

```
1. #encoding=utf8
2. from sklearn.linear_model import LogisticRegression
3.
4. def cancer_clf(train_data,train_label,test_data):
5.     '''
6.     train_data(ndarray):训练数据
7.     train_label(ndarray):训练标签
8.     test_data(ndarray):ces 数据
9.     '''
10.    #***** Begin *****#
11.
12.    #***** End *****#
13.    return predict
```

四、实验报告要求

参见实验报告模板

实验二：机器学习3-k 均值算法

一、实验目的

- 1、理解并掌握 k 均值无监督分类算法思想。
- 2、了解 Mini Batch KMeans 方法对 K-Means 的改进，并实现两种算法的可视化展示。

二、实验内容

- 1、补全一个实现 k 近邻分类方法的小程序。
- 2、实现 K-Means 与 Mini Batch KMeans 方法并进行可视化展示。

三、实验内容与步骤

- 1、数据集准备：6 种不同的聚类数据集

准备好六种不同的聚类数据集，代码如下：

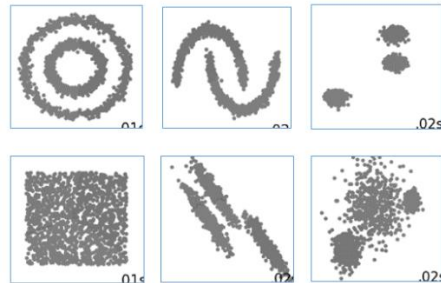
```
1.  from sklearn.cluster import MiniBatchKMeans
2.  from sklearn.cluster import KMeans
3.  import numpy as np
4.
5.  X = np.array([[1,2],[1,4],[1,0],
6.               [4,2],[4,0],[4,4],
7.               [4,5],[0,1],[2,2],
8.               [3,2],[5,5],[1,-1]])
9.  n = int(input())
10. if n== 0:
11.     #MiniBatchKMeans 模块
12.     #***** Begin *****#
13.
14.
15.     #***** End *****#
16.
17. else:
18.     #KMeans 模块
19.     #***** Begin *****#
20.
21.     #***** End *****#
22.
23.     #输出所有点的类别、两类的中心点并预测[0,0],[4,4]的类别
```

```

24. ##### Begin #####
25.
26. ##### End #####

```

将会得到如下图所示的六类数据集：



2、设置聚类参数。设置聚类参数代码如下：

```

1. default_base = {'quantile': .3,
2.                 'eps': .3,
3.                 'damping': .9,
4.                 'preference': -200,
5.                 'n_neighbors': 10,
6.                 'n_clusters': 3}
7. datasets = [
8.     (noisy_circles, {'damping': .77, 'preference': -240,
9.                     'quantile': .2, 'n_clusters': 2}),
10.    (noisy_moons, {'damping': .75, 'preference': -220, 'n_clusters': 2}),
11.    (varied, {'eps': .18, 'n_neighbors': 2}),
12.    (aniso, {'eps': .15, 'n_neighbors': 2}),
13.    (blobs, {}),
14.    (no_structure, {})]

```

3、创建聚类对象 & 应用聚类方法。创建聚类对象代码如下：

```

1. kmeans = cluster.KMeans(n_clusters=params['n_clusters'])
2. two_means = cluster.MinibatchKMeans(n_clusters=params['n_clusters'])

```

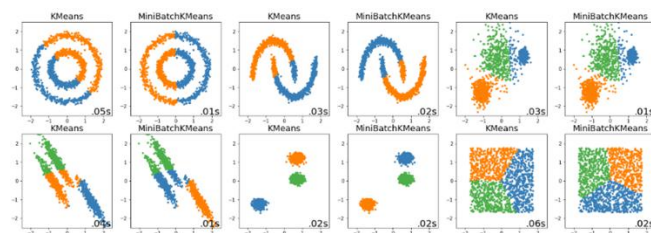
应用聚类方法代码如下：

```

1. for name, algorithm in clustering_algorithms:
2.     t0 = time.time() #start time
3.     algorithm.fit(X) #clustering
4.     t1 = time.time() #end time
5.     y_pred = algorithm.predict(X)

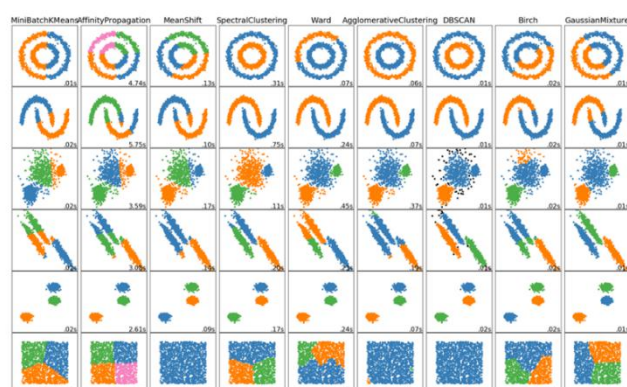
```

聚类结果展示



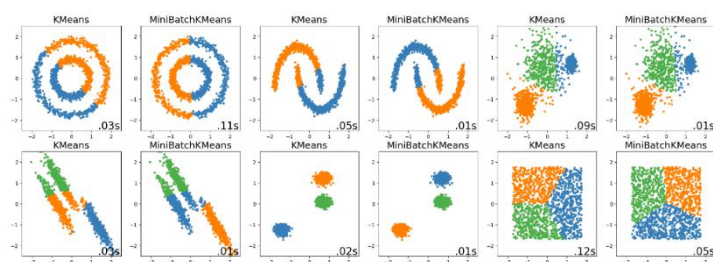
4、其他聚类方法

如谱聚类、DBSCAN、均值移动和自底向上的聚类等方法也需要了解。部分聚类实验结果可视化展示如下：



编程要求：请仔细阅读右侧代码，结合相关知识，在 **Begin-End** 区域内进行代码补充，实现 K-Means 与 Mini Batch KMeans 方法并进行可视化展示。除补充完整代码之外，还应逐行理解右边示例代码，学习各函数及参数的用法，举一反三，运用到其他的分类问题之中。

测试说明：平台会对你编写的代码进行测试。图片预期输出结果为：



四、实验报告要求

参见实验报告模板