分类算法

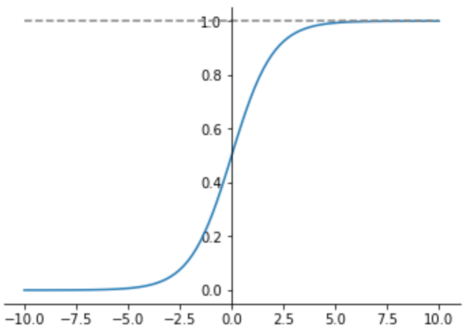
Logistic Regression

- Logistic Regression 事实上是广义线性回归，将普通线性回归（OLS）的结果，对应到二元分类里面

模型

- 考虑到函数性质（可微），我们选取sigmoid函数将OLS结果转化成｛0，1｝变量。sigmoid函数图像（代码见附录）如下所示:

$$y = \frac{1}{1+e^{-x}}$$



* 从而，我们令

$$\text{样本方程：} h_i = \frac{1}{1+e^{X_i^T \beta}} \Rightarrow \ln \frac{h_i}{1-h_i} = X_i^T \beta$$

$$\text{总体方程：} H = \frac{1}{1+e^{X\beta}}$$

其中:

$$X = \begin{bmatrix} 1 & x_1^1 & x_1^2 & \cdots & x_1^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n^1 & x_n^2 & \cdots & x_n^d \end{bmatrix} \qquad X_i = \begin{bmatrix} 1 & x_i^1 & x_i^2 & \cdots & x_i^d \end{bmatrix}^T$$

$$Y = \begin{bmatrix} y_1 & y_2 & y_3 & \cdots & y_n \end{bmatrix}^T \qquad \beta = \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 & \cdots & \beta_n \end{bmatrix}^T$$

如果我们把h_i看成是第i个样本分类为1的概率的话，这个模型会得到很好的理论解释。

梯度下降法

- 损失函数（Loss Function）：
  应用MLE，我们有似然函数:

$$L(\beta) = \prod_{i=1}^{n} h_i^{y_i} (1-h_i)^{1-y_i}$$

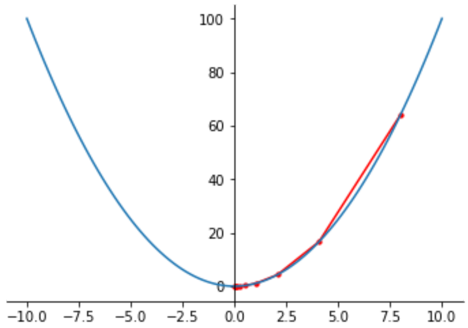
  因此，有损失函数:

$$l(\beta) = -\frac{1}{n} \sum_{i=1}^{n} [y_i \ln(h_i) + (1-y_i) \ln(1-h_i)]$$

从而，我们的问题变成了：

$$\hat{\beta} = arg \min_{\beta} l(\beta)$$

- 梯度下降法（Gradient Descend）：
    1. 梯度（Gradient）
    简单来说，梯度是函数导数的**反方向**。也就是说，梯度是一个**向量**。
    2. Idea
    GD是求解***凸优化问题***的一种方法。比如函数$y = x^2$,给定任一点x_0,从这点出发，沿着梯度方向走，随着走的步数越来越多，其对应的函数值就越接近其最值。如图所示（代码见附录）：



    3. 算法实现
    ① 任意x_0，计算梯度$d_0 = -\frac{\partial f}{\partial x}|_{x=x0}$
    ② 选择步长（学习率）α，更新公式为$x_1 = x_0 + \alpha d_0$
    ③ 以此类推，可以通过判定阀值ϵ，要求$|\ f(x_{k+1}) - f(x_k)\ | \leq \epsilon$;
    或者，选择设定最大迭代次数k，来停止迭代。
    4. 用GD求解Logistic Regression
    ① diff $l(\beta)$ w.r.t. β :

$$\frac{\partial l}{\partial \beta} = -\frac{1}{n}\sum_{i=1}^{n} X_i^T(y_i - h_i) = -\frac{1}{n}X^T(y-h)$$

    ② update:$\beta = \beta + \alpha\frac{\partial l}{\partial \beta}$

算法实现

- 导入数据集
  #这个note的所有数据集都可以在我的GitHub主页找到

```
import numpy as np


def load_data(path):
    """ load data from txt file
    this function requires data structure to be [features, label]

    input:  path(str): the path of data file
    output: label(mat): an n*1 matrix of label
            features(mat): a n*(d+1) matrix of features
    """
    lines = []
    with open(path) as f:
        for line in f.readlines():
            lines.append(line.split())
    raw_data = np.array(lines, dtype=float)

    n = raw_data.shape[0]
    label = raw_data[:, -1].reshape((n, 1))
    features = raw_data.copy()
    features[:, -1] = 1

    return np.mat(label), np.mat(features)
```

- 定义sigmoid函数，并实现梯度下降

```python
def sig(x):
    return 1 / (1 + np.exp(-x))


def logit_gd(features, label, max_cycle, step):
    """train Logistic model with Gradient Descend

    input:  features(mat): a n*(d+1) matrix of features
            label(mat): an n*1 matrix of label
            max_cycle(int): maximum iteration times
            step(float): learning ratio
    output: beta(mat): a (d+1)*1 matrix of parameters
    """
    n, d = features.shape
    # initialize beta
    beta = np.ones((d,1))

    while max_cycle:
        max_cycle -= 1
        # calculate the gradient
        err = label - sig(features * beta)
        gd = features.T * err

        # track the approximate error (not necessary)
        if max_cycle % 100 == 0:
            error = np.sum(err) /n
            print("error: {} {}".format(max_cycle, error))

        # update beta
        beta += step * gd

    return beta
```

- 计算预测值和准确率

```python
def get_predict(features, beta):
    """predict label uses trained model
    input:  features(mat): a n*(d+1) matrix of features
            beta(mat): a (d+1)*1 matrix of parameters
    output: prediction(mat): a n * 1 predicted value of label
    """
    h = sig(features * beta)
    n = features.shape[0]
    predict_label = []
    for i in range(n):
        if h[i,0]>0.5:
            predict_label.append(1)
            continue
        predict_label.append(0)
    prediction = np.array(predict_label).reshape((n,1))

    return prediction

def get_accuracy(label, prediction):
    """calculate the accuracy of the model
    input:  label(mat): an n*1 matrix of label
            prediction(mat): a n * 1 predicted value of label
    output: acc(float): accuracy of the model
    """
    n = label.shape[0]
    result = 0
    for i in range(n):
        if label[i,0] == prediction[i,0]:
            result += 1
    acc = result / n
    return acc
```

- 运行

```python
if __name__ == "__main__":
    label, features = load_data("data/1.logit_data.txt")
    beta = logit_gd(features, label, 1000, 0.01)
    prediction = get_predict(features,beta)
    accuracy = get_accuracy(label,prediction)
    print(beta)
```

附录1

- sigmoid 图像绘制

```python
import numpy as np
import matplotlib.pyplot as plt


def sig(x):
    return 1/(1 + np.exp(-x))


x = np.linspace(-10, 10, 100)
y = np.ones((100,))
plt.plot(x, sig(x))
plt.plot(x, y, c='grey', linestyle='--')
ax = plt.gca()
ax.spines['top'].set_visible(False)  # 去掉上边框
ax.spines['right'].set_visible(False)  # 去掉右边框
ax.spines['left'].set_position(('data', 0))  # 移动左坐标轴到数据为0的位置
```

- GD图示

```python
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return x**2
x = np.linspace(-10, 10, 100)

step = 10
x0 = 8
while step>0:
    step -= 1
    x1 = x0 - 2 * x0*0.245
    plt.scatter(x0,f(x0),c="r",s=10)
    plt.plot([x0,x1],[f(x0),f(x1)],"r")
    x0 = x1
plt.plot(x,f(x))

ax = plt.gca()
ax.spines['top'].set_visible(False)  # 去掉上边框
ax.spines['right'].set_visible(False)  # 去掉右边框
ax.spines['left'].set_position(('data', 0))  # 移动左坐标轴到数据为0的位置
```