

数据分析及实践-实验四

PB20000326 徐海阳

数据分析及实践-实验四

Part 1 分类算法实践

1.1 算法主要流程

1.2 算法关键技术

1.3 算法实现

1.4 实验记录

Part 2 预测算法实践

2.1 lightgbm

附录

part 1完整代码及实现细节

Part 1 分类算法实践

1.1 算法主要流程

Step 1. 将lab3中特征工程得到的DIY.csv转换为.npy文件，并进行5折交叉验证所需要的train/test数据集划分

```
1 data = np.load('./data/DIY.npy')
2
3 data_part = cross_validation(data, 5)
4
5 for i in range(5):
6     print('part: %d' % (i+1))
7     train, test = split_train_test(data_part, i)
```

Step 2. 用logistic回归进行二分类。

```
1 class MLP(object):
2     # ...
3     for i in range(5):
4         print('part: %d' % (i+1))
5         train, test = split_train_test(data_part, i)
6         model = MLP(train, test, train.shape[1]-1, 1, 0.1, 5001)
7         model.train_model()
```

1.2 算法关键技术

Step 1: 将DIY.npy数据分为5个part，其中四个concatenate成为train，剩余一个为test。

Step 2: 构造class: MLP，实现sigmoid，sigmoid_derivative（对sigmoid求导），forward（前向传播），judge（judge>0则代表predict结果正确），backward（反向传播），test_model（在test集上计算正确率），train_model（在train集上计算正确率）方法。

1.3 算法实现

关键代码结构如下。完整代码见附录。

```
1  import numpy as np
2
3  def cross_validation(data, k):
4      # split the data into k parts
5      # return the k parts
6      return data.reshape(k, -1, data.shape[1])
7
8  def split_train_test(data_part, k):
9      # split the data_part into train and test
10     # return the train and test
11     test = data_part[k, :, :]
12     train = np.delete(data_part, k, axis=0)
13     train = np.concatenate(train, axis=0)
14     return train, test
15
16 class MLP(object):
17     def __init__(self, train, test, n_in, n_out, lr, epoch):
18         # ...
19
20     def sigmoid(self, x):
21         # ...
22
23     def sigmoid_derivative(self, x):
24         # ...
25
26     def forward(self, x): # (bsz, n_in)
27         # ...
28
29     def judge(self, y_hat, y):
30         """
31         if (y_hat > 0.5 and y = 1) or (y_hat < 0.5 and y = 0), then return positive:
32         right predict!
33         create a function to judge whether the prediction is correct
34         """
35         # ...
36
37     def backward(self, x, y): # (bsz, n_in) , (bsz, n_out)
38         # ...
39
40     def test_model(self, data, label):
41         # ...
42
43     def train_model(self):
44         # ...
45
46 def main():
47     data = np.load('./data/DIY.npy')
48
49     data_part = cross_validation(data, 5)
50
51     for i in range(5):
52         print('part: %d' % (i+1))
```

```

53     train, test = split_train_test(data_part, i)
54     model = MLP(train, test, train.shape[1]-1, 1, 0.1, 5001)
55     model.train_model()
56     print('\n\n')
57
58 if __name__ == '__main__':
59     main()

```

1.4 实验记录

(k折交叉验证, 4:1比例, 共有5折)

在learning_rate=0.1, epoch=5001时的结果如下:

```

C:\Users\x\Desktop\twodown\USTC_AD2022_Lab\lab4>python lab4_part1.py
part: 1
epoch: 5000, accs: 0.917989

part: 2
epoch: 5000, accs: 0.929038

part: 3
epoch: 5000, accs: 0.922502

part: 4
epoch: 5000, accs: 0.931684

part: 5
epoch: 5000, accs: 0.929194

```

k	ACC
1	0.917989
2	0.929038
3	0.922502
4	0.931684
5	0.929194
平均值	0.926081

Part 2 预测算法实践

1. 尝试了lightgbm库
2. 尝试了sklearn库中的12种回归算法

2.1 lightgbm

附录

part 1完整代码及实现细节

```
1 import numpy as np
2
3 def cross_validation(data, k):
4     # split the data into k parts
5     # return the k parts
6     return data.reshape(k, -1, data.shape[1])
7
8 def split_train_test(data_part, k):
9     # split the data_part into train and test
10    # return the train and test
11    test = data_part[k, :, :]
12    train = np.delete(data_part, k, axis=0)
13    train = np.concatenate(train, axis=0)
14    return train, test
15
16 class MLP(object):
17     def __init__(self, train, test, n_in, n_out, lr, epoch):
18         # self.bsz = train.shape[0]
19         # self.train = train
20         # self.test = test
21         self.n_in = n_in
22         self.n_out = n_out
23         self.lr = lr
24         self.epoch = epoch
25         self.train_data = train[:, :-1]
26         train_label = train[:, -1]
27         self.train_label = train_label.reshape(train_label.shape[0], 1)
28         self.test_data = test[:, :-1]
29         test_label = test[:, -1]
30         self.test_label = test_label.reshape(test_label.shape[0], 1)
31         self.w = np.random.randn(self.n_in, self.n_out)
32         self.b = np.random.randn(self.n_out)
33
34     def sigmoid(self, x):
35         return 1 / (1 + np.exp(-x))
36
37     def sigmoid_derivative(self, x):
38         return self.sigmoid(x) * (1 - self.sigmoid(x))
39
40     def forward(self, x): # (bsz, n_in)
41         o = np.dot(x, self.w) + self.b # (bsz, n_out)
42         y_hat = self.sigmoid(o) # (bsz, n_out)
43         return o, y_hat
44
45     def judge(self, y_hat, y):
```

```

46         """
47         if (y_hat > 0.5 and y = 1) or (y_hat < 0.5 and y = 0),then return positive:
right predict!
48         create a function to judge whether the prediction is correct
49         """
50         s = (y_hat - 0.5) * (y - 0.5)
51         return s
52
53     def backward(self, x, y): # (bsz, n_in) , (bsz, n_out)
54         bsz = x.shape[0]
55         o, y_hat = self.forward(x) # (bsz, n_out)
56
57         # 链式法则
58         d_L_d_y_hat = -y/y_hat + (np.ones_like(y)-y)/(np.ones_like(y)-y_hat) #
(bsz, n_out)
59         d_y_hat_d_o = self.sigmoid_derivative(y_hat) # (bsz, n_out)
60         d_o_d_w = x # (bsz, n_in)
61         d_o_d_b = np.ones((bsz, 1)) # (bsz, 1)
62
63         d_L_d_w = np.mean(d_L_d_y_hat * d_y_hat_d_o * d_o_d_w, axis=0) # (n_in,)
64         d_L_d_w = d_L_d_w.reshape(self.n_in, 1) # (n_in, 1)
65         d_L_d_b = np.mean(d_L_d_y_hat * d_y_hat_d_o * d_o_d_b, axis=0) # (1,)
66         self.w = self.w - self.lr * d_L_d_w
67         self.b = self.b - self.lr * d_L_d_b
68
69
70     def test_model(self, data, label):
71         # pdb.set_trace()
72         o, y_hat = self.forward(data)
73         total = data.shape[0]
74         """
75         correct are those whose prediction is correct, i.e. y_hat > 0.5 and y = 1
76         """
77         correct = np.sum(self.judge(y_hat, label)>0)
78         accs = correct / total
79         return accs
80
81     def train_model(self):
82         for i in range(self.epoch):
83             self.backward(self.train_data, self.train_label)
84             if i!=0 and i%5000 == 0:
85                 accs = self.test_model(self.test_data, self.test_label)
86                 print('epoch: %d, accs: %f' % (i, accs))
87
88 def main():
89     data = np.load('./data/DIY.npy')
90
91     data_part = cross_validation(data, 5)
92
93     for i in range(5):
94         print('part: %d' % (i+1))
95         train, test = split_train_test(data_part, i)
96         model = MLP(train, test, train.shape[1]-1, 1, 0.1, 5001)
97         model.train_model()
98         print('\n\n')
99
100 if __name__ == '__main__':
101     main()

```

