zfmlhw04

1. 阅读题

在过去的两节课中，你已经对Logistic Regression的理论进行了较为深入的学习。Logistic Regression在实践中也有很多具体的应用。例如在自然语言处理（NLP）领域，你可以利用Logistic Regression构建分类器，实现对客户评价的情感偏向分析。假设一名观众在观看了一部电影之后写下了一条评语，那么你所构建的分类器结合NLP的技术就可以用来判别该名观众是否喜欢这部电影。下面这篇博文演示了利用Logistic Regression来实现一个最为简单的客户情感偏向分析应用的基本思路及方法：
http://blog.csdn.net/baimafujinji/article/details/51153872
(http://blog.csdn.net/baimafujinji/article/details/51153872)
为了帮助你构建“如何应用逻辑回归来解决实际问题”这样一个具体的认识，请你阅读这篇博文，并提交一张你阅读的界面截图。学习是自己的事情，请认真理解其中内容，切勿只截图而不阅读。

原 **自然语言处理实战之微博情感偏向分析**

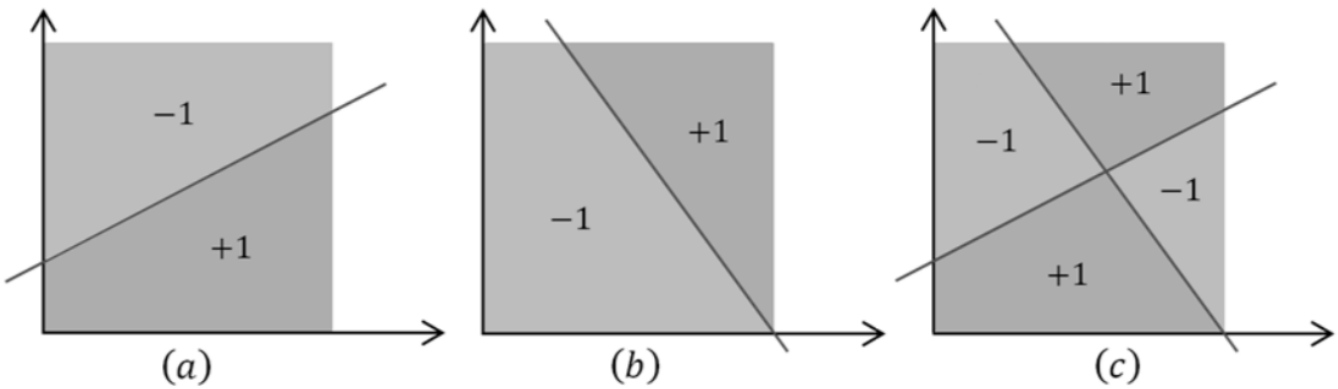
2016年04月14日 23:22:38 白马负金羁 阅读数：11940 标签： NLP Scikit-Learn Sentiment Analysis 微博情感分析 更多

自然语言处理（NLP）中一个很重要的研究方向就是语义的情感分析（Sentiment Analysis）。例如IMDB上有很多关于电影的评论，那么我们就可以通过Sentiment Analysis来评估某部电影的口碑，（如果它才刚刚上映的话）甚至还可以据此预测它是否能够卖座。与此相类似，国内的豆瓣上也有很多对影视作品或者书籍的评论内容亦可以作为情感分析的语料库。对于那些电子商务网站而言，针对某一件商品，我们也可以看到留言区里为数众多的评价内容，那么同类商品中，哪个产品最受消费者喜爱呢？或许对商品评论的情感分析可以告诉我们答案。

在之前的文章中，笔者已经向读者介绍了在Python中利用NLTK进行自然语言处理的一些基本方法：

2. 问答题

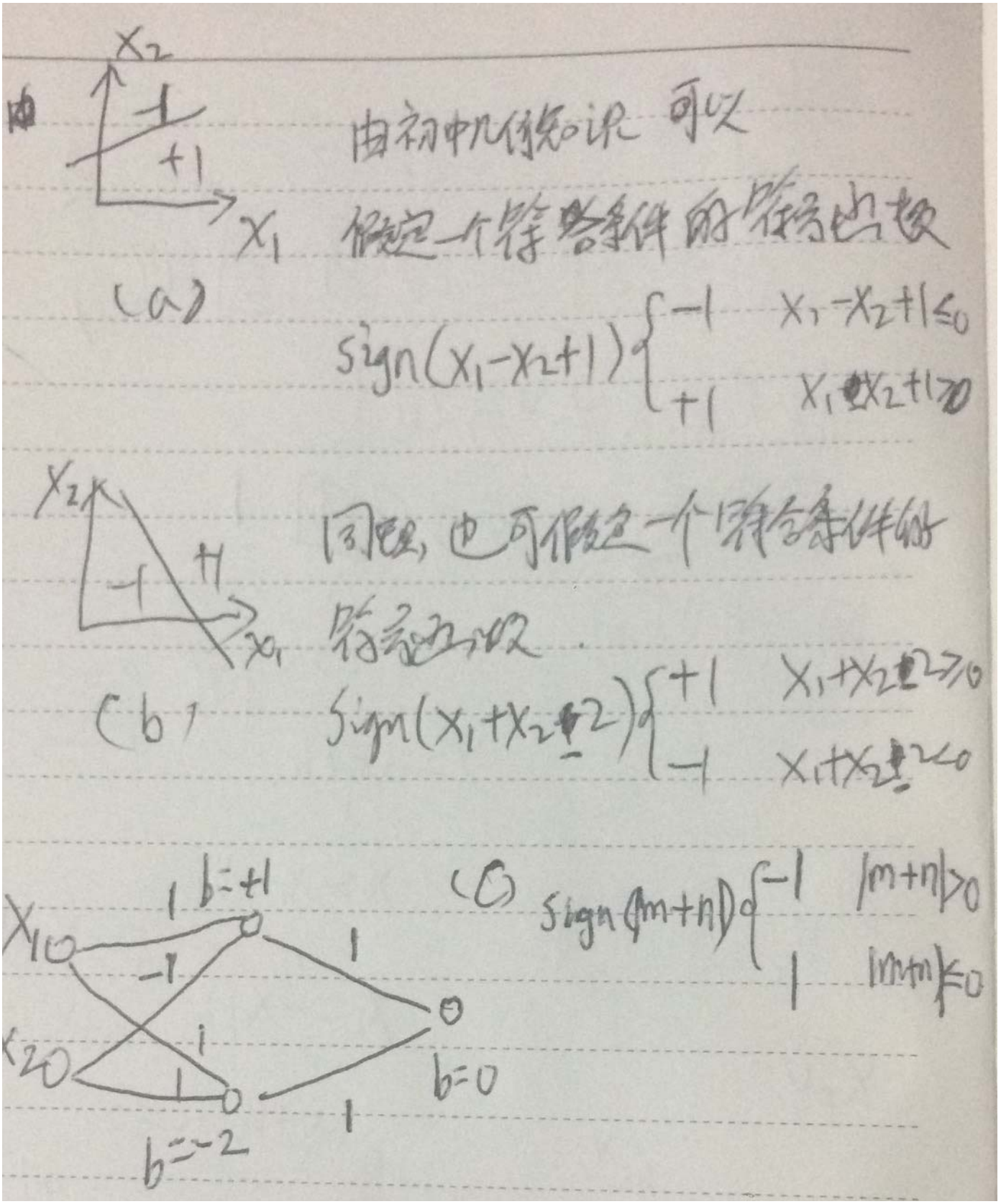
俗话说“三个臭皮匠赛过诸葛亮”，感知机的强大之处就在于：将若干简单的感知机结合起来便可以完成更加复杂甚至原本无法完成的任务。例如，下图中，通过将(a)和(b)这两个感知机组合起来，就可以实现如图(c)所示的异或运算。



请你设计能够实现上述XOR运算的多层感知机。提示：根据离散数学知识可得

$$XOR(g_1, g_2) = (\neg g_1 \wedge g_2) \vee (g_1 \wedge \neg g_2)$$

因此，你可以先做一层交运算，再做一层并运算。并注意交运算中隐含有一层取反运算。
请你画出并提交完整的多层感知机模型图，注意标清楚诸如权重、偏置、激励函数等信息。



3. 实践题

在课堂上我们演示了如何利用TensorFlow并基于Softmax（或称多元逻辑回归）来进行手写数字识别的例子（基于MNIST数据集）。下面请你在不使用TensorFlow的情况下来完成这个任务（同样是基于多元逻辑回归）。
具体要求与提示：
a）编程语言上，你可以使用Python或者R。
b）如果你使用的是Python，那么你可以使用 Scikit-Learn中的LogisticRegression（或者LogisticRegressionCV），但你不可以使用TensorFlow，否则你会被红牌扣罚。
关于LogisticRegression你可以参考：http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)
关于LogisticRegressionCV你可以参考：http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegressionCV.html (http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegressionCV.html)
c）如果你使用的是R，那么你可以使用maxent包，但不可以使用softmaxreg包，否则你会被红牌扣罚。
关于maxent包，你可以参考https://cran.r-project.org/web/packages/maxent/maxent.pdf (https://cran.r-project.org/web/packages/maxent/maxent.pdf)
d）获取MNIST数据的方法：从http://yann.lecun.com/exdb/mnist/ (http://yann.lecun.com/exdb/mnist/) 上直接下载。但是，如果你选择使用
Python，你也可以参考： http://scikit-learn.org/stable/datasets/index.html (http://scikit-learn.org/stable/datasets/index.html) （在页面上搜索MNIST，可以看到相关方法）。
e）你需要利用MNIST中提供的test数据集来测试你构建的分类器，并提供测试结果的截图。
f）你需要提交完整的、可以正确运行的源代码（包含引用必要头文件所需的代码）。

In [14]:

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
```

In [5]:

```
mnist = datasets.fetch_mldata('MNIST original')
mnist
```

Out[5]:

```
{'DESCR': 'mldata.org dataset: mnist-original',
 'COL_NAMES': ['label', 'data'],
 'target': array([0., 0., 0., ..., 9., 9., 9.]),
 'data': array([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)}
```

In [6]:

```
mnist.data.shape
```

Out[6]:

```
(70000, 784)
```

In [9]:

```
X_train, X_test, y_train, y_test = train_test_split(mnist.data, mnist.target, test_
```

In [10]:

```
lm = LogisticRegression(C=1e5,  solver='lbfgs', multi_class='multinomial')
```

In [11]:

```
lm.fit(X_train, y_train)
```

Out[11]:

```
LogisticRegression(C=100000.0, class_weight=None, dual=False,
          fit_intercept=True, intercept_scaling=1, max_iter=100,
          multi_class='multinomial', n_jobs=1, penalty='l2',
          random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
          warm_start=False)
```

In [12]:

```
prediction = lm.predict(X_test)
```

In [15]:

```
print("accuracy score: ")
print(accuracy_score(y_test, prediction))
print(classification_report(y_test, prediction))
```

```
accuracy score:
0.9210714285714285
             precision    recall  f1-score   support

        0.0       0.95      0.96      0.95      1349
        1.0       0.96      0.98      0.97      1581
        2.0       0.91      0.90      0.90      1400
        3.0       0.91      0.90      0.90      1434
        4.0       0.94      0.93      0.93      1328
        5.0       0.88      0.87      0.88      1286
        6.0       0.95      0.94      0.94      1407
        7.0       0.94      0.93      0.94      1476
        8.0       0.88      0.90      0.89      1391
        9.0       0.88      0.90      0.89      1348

avg / total       0.92      0.92      0.92     14000
```

end