1 机器学习导论

姓名: 殷天润 学号:171240565

题目 (ML problem 1)

[20pts] Naive Bayes Classifier

We learned about the naive Bayes classifier using the "property conditional independence hypothesis". Now we have a data set as shown in the following table:

表 1: Dataset

	x_1	x_2	x_3	x_4	y
Instance1	1	1	1	0	1
Instance2	1	1	0	0	0
Instance3	0	0	1	1	0
Instance4	1	0	1	1	1
Instance5	0	0	1	1	1

(1) [10pts] Calculate: $\Pr\{y = 1|\mathbf{x} = (1, 1, 0, 1)\}$ and $\Pr\{y = 0|\mathbf{x} = (1, 1, 0, 1)\}$.

(2) [10pts] After using Laplacian Correction, recalculate the value in the previous question.

解答:

1. (a)
   - P(y=1)=$\frac{3}{5}$=0.6
   - P(y=0)=$\frac{2}{5}$=0.4

   (b)
   - P($x_1 = 1|y = 1$)=$\frac{2}{3}$
   - P($x_1 = 1|y = 0$)=$\frac{1}{2}$

   (c)
   - P($x_2 = 1|y = 1$)=$\frac{1}{3}$
   - P($x_2 = 1|y = 0$)=$\frac{1}{2}$

   (d)
   - P($x_3 = 0|y = 1$)=0
   - P($x_3 = 0|y = 0$)=$\frac{1}{2}$

   (e)
   - P($x_4 = 1|y = 1$)=$\frac{2}{3}$
   - P($x_4 = 1|y = 0$)=$\frac{1}{2}$

   所以:

   - $Pr\{y = 1|\mathbf{x} = (1, 1, 0, 1)\} = P(y = 1) \times P(x_1 = 1|y = 1) \times P(x_2 = 1|y = 1) \times P(x_3 = 0|y = 1) \times P(x_4 = 1|y = 1) = 0$
   - $Pr\{y = 0|\mathbf{x} = (1, 1, 0, 1)\} = P(y = 0) \times P(x_1 = 1|y = 0) \times P(x_2 = 1|y = 0) \times P(x_3 = 0|y = 0) \times P(x_4 = 1|y = 0) = 0.025$

2. 使用拉普拉斯修正:

(a)
- P(y=1)=$\frac{3+1}{5+2}$=0.57142
- P(y=0)=$\frac{2+1}{5+2}$=0.42857

(b)
- P($x_1 = 1|y = 1$)=$\frac{2+1}{3+2}$=0.6
- P($x_1 = 1|y = 0$)=$\frac{1+1}{2+2}$=0.5

(c)
- P($x_2 = 1|y = 1$)=$\frac{1+1}{3+2}$=0.4
- P($x_2 = 1|y = 0$)=$\frac{1+1}{2+2}$=0.5

(d)
- P($x_3 = 0|y = 1$)=$\frac{0+1}{3+2}$=0.2
- P($x_3 = 0|y = 0$)=$\frac{1+1}{2+2}$=0.5

(e)
- P($x_4 = 1|y = 1$)=$\frac{2+1}{3+2}$=0.6
- P($x_4 = 1|y = 0$)=$\frac{1+1}{2+2}$=0.5

所以:

- $Pr\{y = 1|\mathbf{x} = (1, 1, 0, 1)\} = P(y = 1) \times P(x_1 = 1|y = 1) \times P(x_2 = 1|y = 1) \times P(x_3 = 0|y = 1) \times P(x_4 = 1|y = 1) = 0.016457$
- $Pr\{y = 0|\mathbf{x} = (1, 1, 0, 1)\} = P(y = 0) \times P(x_1 = 1|y = 0) \times P(x_2 = 1|y = 0) \times P(x_3 = 0|y = 0) \times P(x_4 = 1|y = 0) = 0.026785$

题目 (ML problem 2)

[20pts] Bayes Optimal Classifier

For a binary classification task, when data in the two classes satisfies Gauss distribution and have the same variance, please prove that LDA can produce the bayes optimal classifier.

解答:

贝叶斯最优分类器满足 $h^*(x) = arg\ max_{c\in y} Pr(c|x)$。根据贝叶斯定理, 有 $h^*(x) = arg\ max_{c\in y} P(c|x)$

现在已知协方差矩阵 $\sum$, 根据已知条件, 贝叶斯最优分类器可以表示为:

$$\begin{aligned}
h^*(x) &= arg\ max P(h = c)|X = x) \\
&= arg\ max f_c(x)P(c) \\
&= arg\ max_c\ log(f_c(x)P(c)) \\
&= arg\ max_c\ log[\frac{1}{(2\pi)^{d/2}|\sum|^{1/2}} * e^{-\frac{1}{2}(x-\mu_c)^T * \sum^{-1} * (x-\mu_c)} \\
&= arg\ max_c[-log((2\pi)^{d/2} * |\sum|^{0.5}) - \frac{1}{2}(x - \mu_c) + log(P(c))] \\
&= arg\ max_c[-0.5 * (x - \mu_c)^T * (\sum)^{-1}(x - \mu^c) + log(P(c))]
\end{aligned}$$

而:

$$\frac{-1}{2}(x - \mu_c)^T * (\sum)^{-1}(x - \mu_c) = x^T(\sum)^{-1}\mu_c - \frac{1}{2}\mu_c^T(\sum)^{-1}\mu_c - \frac{1}{2}x^T(\sum)^{-1}x$$

所以

$$h^*(x) = arg\ max_c[x^T(\sum)^{-1}\mu_c - 0.5\mu_c^T(\sum)^{-1}\mu_c + log(P(c))]$$

这就是 LDA 最优分类器;

题目 (ML problem 3)

[60pts] Ensemble Methods in Practice

Due to their outstanding performance and robustness, ensemble methods are very popular in machine community. In this experiment we will practice ensemble learning methods based on two classic ideas: Boosting and Bagging.

In this experiment, we use an UCI dataset Adult. You can refer to the link[1] to see the data description and download the dataset.

Adult is an class imbalanced dataset, so we select AUC as the performance measure. You can adopt sklearn to calculate AUC.

(1) [10pts] You need finish the code in Python, and only have two files: AdaBoost.py, RandomForestMain.py. (The training and testing process are implemented in one file for each algorithm.)

(2) [40pts] The is experiment requires to finish the following methods:

- Implement AdaBoost algorithm according to the Fig(8.3), and adopt decision tree as the base learner (For the base learner, you can import sklearn.)
- Implement Random Forest algorithm. Please give a pseudo-code in the experiment report.
- According to the AdaBoost and random forest, analysis the effect of the number of base learners on the performance. Specifically, given the number of base learners, use 5-fold cross validation to obtain the AUC. The range of the number of base learners is decided by yourself.
- Select the best number of base classifiers for AdaBoost and random forests, and obtain the AUC in the test set.

(3) [10pts] In the experimental report, you need to present the detail experimental process. The experimental report needs to be hierarchical and organized, so that the reader can understand the purpose, process and result of the experiment.

解答:

1. 我完成了两份 python 代码, 但是在数据提取上, 我需要额外对 adult.test 数据集的 label 做一些额外的工作——把最后一个'.' 去掉。我现在使用的是经过上述操作之后的 adult2.test 作为数据集输入;

[1] http://archive.ics.uci.edu/ml/datasets/Adult

2. 下面是我的实验报告, 主要分为:1. Random Forest algorithm 的伪代码, 并且解释我的 Random Forest 代码;2. 解释我的 Adaboost 代码;3. 解释我取数据,5-折交叉验证数据部分的代码;4. 两个算法中基学习器和 AUC 的关系;5. 在实验中遇到的 bug 以及困难;

1.1 Random Forest 的伪码:

Algorithm 1 Random Forest

输入

训练集 D=$\{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$

基学习算法 $\mathcal{L}$S

X 中的特征:Features

树的数量:$n\_estimators$

过程

sub_set=random(D) 随机化数据集

for $t = 1, 2, ..., n\_estimators$ **do**

temp_features=random(Features) 从 Features 随机选取特征;

X,Y=sub_set(temp_features) 根据 features 从 sub_set 里面构造数据集;

Tree(t)=$\mathcal{L}(X, Y)$ 通过 X,Y 训练决策树;

features(t)=temp_features 存下该树的特征;

end for

输出

R(x)=mean(Tree,features) 通过训练出来的 Tree 以及 features 的平均值得到输出;

1.2 对我自己的代码的解释:

1.2.1 数据提取部分: 代码提取部分主要是从 adult.data 以及 adult2.test 里面

```
def get_data():
    #adult_data.shape
    #adult2.test的数据是adult的数据删去label的最后一个字符"."得到的;
    adult_header=["age","workclass","fnlwgt","education","education-num",
              "marital-status","occupation","relationship","race","sex","capital-gain",
              "capital-loss","hours-per-week","native-country","label"]
    adult_data=pd.read_csv("./adult.data",index_col=False,names=adult_header)
    adult_test=pd.read_csv("./adult2.test",index_col=False,names=adult_header)
    #处理带有?的项目;
    adult_data[adult_data==" ?"]=np.nan
    adult_test[adult_test==" ?"]=np.nan
    adult_data.dropna(axis=0,how='any',inplace=True)
    adult_test.dropna(axis=0,how='any',inplace=True)
    #对非数值的数据进行处理;
    discre_name=["workclass","education","marital-status",
             "occupation","relationship","race",
             "sex","native-country","label"]
    for name in discre_name:
        key=np.unique(adult_data[name])
        #print(key)
        le=preprocessing.LabelEncoder()
        le.fit(key)
        adult_test[name]=le.transform(adult_test[name])
        adult_data[name]=le.transform(adult_data[name])
    #合并测试,训练两个数据集(之后用5-折交叉验证找)
    data = np.vstack((adult_data, adult_test))
    X = data[:, 0:-1]
    Y = data[:, -1]
    return X, Y
```

获得数据, 并且通过 dropna 将有” ?” 的数据去掉; 最后返回组合好了的 X,Y 以供后面的函数调用进行 5 折交叉验证;

1.2.2 Random forest 类:

- 参数:
  n_estimators=0 # 树的数量
  max_features=0 # 每棵树的选用数据集的最大特征数
  min_samples_split=0 # 每棵树最小分割数
  min_gain=0 # 每一颗树到 min_gain 之后就停止
  max_depth=0 # 每一颗树的最大层数
  trees=[ ] # 森林
  trees_feature=[ ] # 用来记录每一个树用了哪些特征
- 函数:
- def ___init___ 是常规的初始化操作;
- def get_bootstrap_data(self,X,Y): 通过有放回的随机化, 随机出一个大小和之前一样大的数据集, 用于下面随机森林的训练;
- def fit(self,X_train,Y_train): 运用伪代码的算法如上图代码所示;

```
def fit(self,X_train,Y_train):
    # 每一颗树都通过get_bookstrap_data获得随机的数据集

    sub_sets=self.get_bootstrap_data(X_train,Y_train)
    n_features=X_train.shape[1]

    if self.max_features == None:
        self.max_features = int(np.sqrt(n_features))

    for i in range (self.n_estimators):
        # 现在为每一颗树选择随机的特征
        tree =DecisionTreeClassifier(min_samples_split=self.min_samples_split
                                     ,min_impurity_decrease = self.min_gain,max_depth=self.max_depth)

        sub_X,sub_Y=sub_sets[i]
        features=np.random.choice(n_features,self.max_features,replace=True)
        sub_X=sub_X[:,features]
        #print("X",sub_X)
        #print("X",sub_Y)
        tree.fit(sub_X,sub_Y)
        self.trees.append(tree)
        self.trees_feature.append(features)
```

- def predict(self,X): 我想了两个方案来通过上面的随机树来获得结果:
  方案一是获得众数, 用出现最多的 (0/1) 直接作为结果;
  方案二是获得平均值, 最后我选择的方案二, 作为最后的答案输出;

```
def predict(self,X):
    y_preds=[]
    for i in range(self.n_estimators):
        features=self.trees_feature[i]
        sub_X=X[:,features]
        y_pre=self.trees[i].predict(sub_X)
        y_preds.append(y_pre)

    y_preds=np.array(y_preds).T
    #print(y_preds)
    y_pred=[]

    for y_p in y_preds:
        #np.bincount()可以统计每个索引出现的次数,np.argmax()可以返回数组中最大值的索引
        #方案一 获得众数
        y_pred.append(np.bincount(y_p.astype('int')).argmax())
    #方案二 获得平均值
    y_pred=np.mean(y_preds,axis=1)
    #print("2",mat(y_pred).shape)
    return y_pred
```

2 对我 adaboost 部分的代码解释;

2.1 数据提取部分和 1.2.1 用的是一样的函数;

2.2 AdaBoost 类:

- 参数:
  T=500 # 用来确定训练的基分类器个数
  weakClassArr=[ ] # 用来存基分类器的 alpha
  weakalpha=[ ] # 用来存基分类器
  max_depth=0 # 用来确定树的最大 depth
- 函数:
- 函数的实现基本和书上的伪代码一致, 唯一有一些区别的是我在 predict 里面返回的是加权之后的小数, 没有用 sgn 函数进行 0/1 化;

3 关于交叉验证部分的代码:

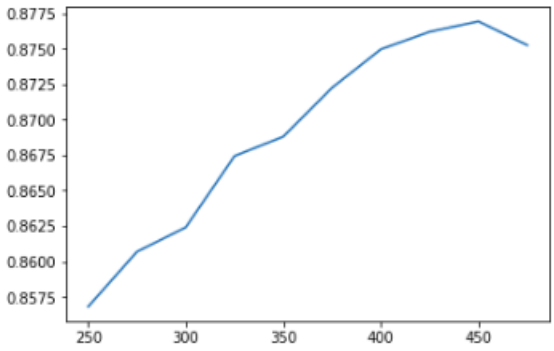
```
X,Y=get_data()
test_num=5
plot_T=[]
plot_auc=[]
for t in range(5,20):
    serand=t
    mean_auc=0.0
#     mean_score=0.0
    for i in range(test_num):
        X_data,X_test,Y_data,Y_test=train_test_split(
                X, Y, test_size=.20, random_state=i*serand)
        Adaboost_classfier=adaboost(t*20,2)
        Adaboost_classfier.fit(X_data,Y_data)
        Y_pred=mat(Adaboost_classfier.predict(X_test))
        #y_pred=mat(Adaboost_classfier.predict(X_test))
        Y_pred.astype(np.int)
        Y_pred=np.array(np.ravel(Y_pred))
        Y_true=np.array(Y_test)
        precision, recall, thresholds = precision_recall_curve(Y_true,Y_pred)
        pr_auc = auc(recall, precision)
        mean_auc+=pr_auc
        Y_pred=np.sign(Y_pred)
#         score=accuracy_score(Y_test, Y_pred)
#         mean_score+=score
    plot_T.append(t*20)
    plot_auc.append(mean_auc/test_num)
    print("Adaboost:T=%d: auc=%f "%(t*20,(mean_auc/test_num)))
```

我手动设置了 range 的范围以及 t 乘以的系数, 对于范围内的 t 我都将做 test_num 次基于 5 折交叉验证的 adaboost 或者 random forest 的训练并且计算平均的 auc; 基于这些平均的 auc 我也得到了相应的图像;

4 基学习器与 AUC 数据的关系, 我根据数据做了相应的图像:

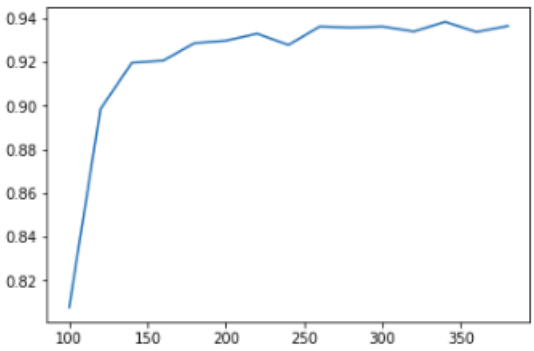
- Randforest:

```
Randomforest: T=250: auc=0.856814
Randomforest: T=275: auc=0.860687
Randomforest: T=300: auc=0.862406
Randomforest: T=325: auc=0.867432
Randomforest: T=350: auc=0.868802
Randomforest: T=375: auc=0.872247
Randomforest: T=400: auc=0.874959
Randomforest: T=425: auc=0.876193
Randomforest: T=450: auc=0.876925
Randomforest: T=475: auc=0.875249
```



- AdaBoost:

```
Adaboost:T=100: auc=0.807721
Adaboost:T=120: auc=0.898425
Adaboost:T=140: auc=0.919583
Adaboost:T=160: auc=0.920561
Adaboost:T=180: auc=0.928580
Adaboost:T=200: auc=0.929699
Adaboost:T=220: auc=0.932903
Adaboost:T=240: auc=0.927747
Adaboost:T=260: auc=0.936189
Adaboost:T=280: auc=0.935687
Adaboost:T=300: auc=0.936173
Adaboost:T=320: auc=0.933890
Adaboost:T=340: auc=0.938313
Adaboost:T=360: auc=0.933670
Adaboost:T=380: auc=0.936414
Out[10]: [<matplotlib.lines.Line2D at 0x7f139dc66198>]
```



5 最后根据上面的分析, 我选择 T=450 作为 Randforest 的参数, 此时的 AUC 为 0.876925; 我选择 T=380 作为 AdaBoost 的参数, 此时 auc=0.936414

6 主要遭遇的 bug 是矩阵计算时候时候的 shape 问题, 以及对 python 各种库函数的不熟悉, 每一次都需要现场找, 导致我花了一整天才完成不涵盖调参的工作;

Hint 现在我提交的 py 文件里面为了减少助教的审核时间用的是比较简单的参数, 可能导致结果和上面不一样, 以及请使用我转化过的 adult2.test 进行代码复现; 我使用 jupyter notebook 写的代码, 原来的代码在两个.ipynb 文件里面, 参数也是报告用的参数;
