AU331 HW1

理论题

1. Solution.

$y = e^{wx+b}$

take logarithm at each side. $\ln y = wx+b$

let $\tilde{y} = \ln y = wx+b = f(x)$

$\Rightarrow$ LSM: $J(w,b) = \sum_{i=1}^{n} (f(x_i) - \tilde{y}_i)^2$

$\Rightarrow J(w^*, b^*) = \min \sum_{i=1}^{n} (wx_i + b - \tilde{y}_i)^2$

$$\begin{cases} \frac{\partial J}{\partial w} = 2\sum_{i=1}^{n} (wx_i + b - \tilde{y}_i) x_i = 0 \\ \frac{\partial J}{\partial b} = 2\sum_{i=1}^{n} (wx_i + b - \tilde{y}_i) = 0 \end{cases}$$

$$\Rightarrow \begin{cases} b^* = \frac{1}{n}\sum_{i=1}^{n} (y_i - wx_i) \\ w^* = \frac{\sum_{i=1}^{n} y_i (x_i - \bar{x})}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2} \end{cases}$$

2. Solution.

随机抽检一个样品,

$$P(次品) = P(A)P(次品|A) + P(B)P(次品|B) + P(C)P(次品|C) = 0.01475$$

根据贝叶斯公式

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$

可得

$$P(A|次品) = \frac{P(次品|A)P(A)}{P(次品)} = \frac{0.015 \times 0.35}{0.01475} = 0.356$$

$$P(B|次品) = \frac{P(次品|B)P(B)}{P(次品)} = \frac{0.010 \times 0.35}{0.01475} = 0.237$$

$$P(C|次品) = \frac{P(次品|C)P(C)}{P(次品)} = \frac{0.020 \times 0.30}{0.01475} = 0.407$$

3. Solution.

对于线性不可分SVM，引入松弛变量 $\xi_i \geq 0$

则 $\min\limits_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i$

$\text{s.t.} \quad y_i(w^T x_i + b) \geq 1-\xi_i$

$\quad\quad \xi_i \geq 0,\ i=1,2,\cdots,m$

故拉格朗日乘子函数 $L(w,b,\alpha,\xi,\mu) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i + \sum_{i=1}^{m}\alpha_i\left(1-\xi_i - y_i(w^T x_i + b)\right) - \sum_{i=1}^{m}\mu_i\xi_i$

$$\begin{cases} \dfrac{\partial L(w,b,\alpha,\xi,\mu)}{\partial w} = w - \sum_{i=1}^{m}\alpha_i y_i x_i \\ \dfrac{\partial L(w,b,\alpha,\xi,\mu)}{\partial b} = -\sum_{i=1}^{m}\alpha_i y_i \\ \dfrac{\partial L(w,b,\alpha,\xi,\mu)}{\partial \xi_i} = C - \sum_{i=1}^{m}\alpha_i - \sum_{i=1}^{m}\mu_i \end{cases}$$

令以上三偏导为0，则有
$$\begin{cases} w = \sum_{i=1}^{m}\alpha_i y_i x_i \\ \sum_{i=1}^{m}\alpha_i y_i = 0 \\ \sum_{i=1}^{m}\alpha_i + \sum_{i=1}^{m}\mu_i = C\cdot M \end{cases}$$

$$g(\alpha,\mu) = \sum_{i=1}^{m}\alpha_i + \frac{1}{2}\|w\|^2 = \sum_{i=1}^{m}\alpha_i + \frac{1}{2}\left(\sum_{i=1}^{m}\alpha_i y_i x_i^T\right)\left(\sum_{i=1}^{m}\alpha_i y_i x_i\right) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j$$

约束条件为 $\left.\begin{array}{l} \alpha_i + \mu_i = C \\ \alpha_i \geq 0,\ \mu_i \geq 0 \end{array}\right\} \Rightarrow 0 \leq \alpha_i \leq C$

故对偶优化问题为 $\max\limits_{\alpha} \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j$

$\text{s.t.} \quad \sum_{i=1}^{m}\alpha_i y_i = 0$

$\quad\quad 0 \leq \alpha_i \leq C$

ξ_i表示第 i 个样本不满足分类约束条件的程度

$\xi_i = 0$，则样本刚好落在最大间隔的边界上

$\xi_i \leq 1$，则样本刚好落在最大间隔的内部

$\xi_i > 1$，则样本被错误分类

实验题一

实验中用到的西瓜 3.0 数据集为 watermelon3.0

对数据集进行预处理后的结果如下

Data Preprocessing

```
In  [84]: samples

Out[84]: array([[1.   , 1.   , 1.   , 1.   , 1.   , 1.   , 0.697, 0.46 ],
                [2.   , 1.   , 2.   , 1.   , 1.   , 1.   , 0.774, 0.376],
                [2.   , 1.   , 1.   , 1.   , 1.   , 1.   , 0.634, 0.264],
                [1.   , 1.   , 2.   , 1.   , 1.   , 1.   , 0.608, 0.318],
                [3.   , 1.   , 1.   , 1.   , 1.   , 1.   , 0.556, 0.215],
                [1.   , 2.   , 1.   , 1.   , 2.   , 2.   , 0.403, 0.237],
                [2.   , 2.   , 1.   , 2.   , 2.   , 2.   , 0.481, 0.149],
                [2.   , 2.   , 1.   , 1.   , 2.   , 1.   , 0.437, 0.211],
                [2.   , 2.   , 2.   , 2.   , 2.   , 1.   , 0.666, 0.091],
                [1.   , 3.   , 3.   , 1.   , 3.   , 2.   , 0.243, 0.267],
                [3.   , 3.   , 3.   , 3.   , 3.   , 1.   , 0.245, 0.057],
                [3.   , 1.   , 1.   , 3.   , 3.   , 2.   , 0.343, 0.099],
                [1.   , 2.   , 1.   , 2.   , 1.   , 1.   , 0.639, 0.161],
                [3.   , 2.   , 2.   , 2.   , 1.   , 1.   , 0.657, 0.198],
                [2.   , 2.   , 1.   , 1.   , 2.   , 2.   , 0.36 , 0.37 ],
                [3.   , 1.   , 1.   , 3.   , 3.   , 1.   , 0.593, 0.042],
                [1.   , 1.   , 2.   , 2.   , 2.   , 1.   , 0.719, 0.103]])

In  [85]: labels

Out[85]: array([1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

1. 实现 LDA 线性分类器并在西瓜 3.0 数据集上用 80%训练、20%测试时的精度

源代码 LDA.ipynb

```
In  [85]: LDA.train(training_samples, training_labels)

In  [86]: pred = LDA.test(testing_samples, testing_labels)

          ACC of test:1.0

In  [87]: pred

Out[87]: array([ 1., -1.,  1.,  1.])

In  [88]: testing_labels

Out[88]: array([ 1., -1.,  1.,  1.])
```

2. 实现 Naïve Bayes 分类器并在西瓜 3.0 数据集上测试 k=5 重交叉验证精度

源代码 NaiveBayes.ipynb

根据 k 重交叉验证的定义，将数据集划分为k份，其中$k-1$份作为训练，1 份作为测试，将k重测试的结果取平均

K=5 重交叉验证的结果如下

```
In  [29]: Cross_validation(samples, labels, flag, k=5)

          ACC of 0th validation : 0.667
          ACC of 1th validation : 1.000
          ACC of 2th validation : 0.667
          ACC of 3th validation : 1.000
          ACC of 4th validation : 0.667

Out[29]: 0.8
```

该部分设计了 k 重交叉验证函数$Cross_validation(samples, labels, flag, k)$

samples 为数据集样本，labels 为标签，flag 是各标签离散/连续情况的标记，0 表示离散特征，1 表示连续特征

k重交叉验证

In [7]:
```
def Cross_validation(samples, labels, flag, k=5):

    batch_size = int(samples.shape[0] / k)
    correct_classification = 0
    total = 0

    for i in range(0, k):
        k_train_samples = np.vstack([samples[0 : i * batch_size], samples[(i + 1) * batch_size :]])
        k_train_labels = np.hstack([labels[0 : i * batch_size], labels[(i + 1) * batch_size:]])

        k_val_samples = samples[i * batch_size : (i + 1) * batch_size]
        k_val_labels = labels[i * batch_size : (i + 1) * batch_size]

        res = NBClassifier(k_train_samples, k_train_labels, k_val_samples, k_val_labels, flag)

        correct_classification += res[1]
        total += res[0]
        print('ACC of %dth validation : %.3f' % (i, res[2]))

    return correct_classification / total
```

3. 比较 SVM 使用不同（至少 4 种）核函数时，西瓜 3.0 数据集上用前 80%训练、后 20%测试的精度

源代码 SVM.ipynb

分别使用 poly, linear, rbf, sigmoid 四种 kernel，结果如下

poly kernel

In [33]:
```
clf = svm.SVC(kernel='poly', C=0.1, degree=3)
clf.fit(training_samples, training_labels)
pred_labels = clf.predict(testing_samples)
print(testing_labels, pred_labels)
print('ACC score: {}'.format(accuracy_score(testing_labels, pred_labels)))
```

```
[ 1.  1. -1.] [ 1.  1. -1.]
ACC score: 1.0
```

linear kernel

In [32]:
```
clf = svm.SVC(kernel='linear', C=0.1)
clf.fit(training_samples, training_labels)
pred_labels = clf.predict(testing_samples)
print(testing_labels, pred_labels)
print('ACC score: {}'.format(accuracy_score(testing_labels, pred_labels)))
```

```
[ 1.  1. -1.] [ 1.  1. -1.]
ACC score: 1.0
```

rbf kernel

In [30]:
```
clf = svm.SVC(kernel='rbf')
clf.fit(training_samples, training_labels)
pred_labels = clf.predict(testing_samples)
print(testing_labels, pred_labels)
print('ACC score: {}'.format(accuracy_score(testing_labels, pred_labels)))
```

```
[ 1.  1. -1.] [ 1.  1. -1.]
ACC score: 1.0
```

sigmoid kernel

In [31]:
```
clf = svm.SVC(kernel='sigmoid')
clf.fit(training_samples, training_labels)
pred_labels = clf.predict(testing_samples)
print(testing_labels, pred_labels)
print('ACC score: {}'.format(accuracy_score(testing_labels, pred_labels)))
```

```
[ 1.  1. -1.] [-1. -1. -1.]
ACC score: 0.3333333333333333
```

实验题二

实现对数几率回归并在西瓜 3.0 和 Iris (http://archive.ics.uci.edu/ml/datasets/Iris) 数据集上与线性分类器、NB 和 SVM 做性能比较

Iris 数据集上的源代码 Iris.ipynb

西瓜 3.0 数据集上的源代码 watermelon.ipynb

实验中使用的 Iris 数据集如下

```
In [25]: Iris_dataset
```

Out[25]:

	0	1	2	3	4
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

对数据集进行预处理，得到 samples 和 labels

```
In [26]: samples
```

```
Out[26]: array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3]
```

Iris 数据集有三个标签 $Iris-setosa, Iris-versicolor, Iris-virginica$，对于二分类的 LDA，需要对数据集进行处理。对于多分类，一般通过$ovo(One\ vs\ One)$或$ovr(One\ vs\ Rest)$将其转化为二分类。

ovr实现如下

One vs Rest

```
In [23]: def ovr(samples, labels):
             for label in set(labels):
                 ones = np.ones_like(labels)
                 zeros = np.zeros_like(labels)
                 new_labels = np.where(labels==label, ones, zeros)
```

LDA 在 Iris 数据集上ovr 5 折交叉验证结果如下

```
In  [119]: Cross_validation(samples, labels, multiclass=True, k=5)

one vs rest for label_0:
ACC of 0th validation : 0.967
ACC of 1th validation : 1.000
ACC of 2th validation : 0.967
ACC of 3th validation : 1.000
ACC of 4th validation : 1.000
one vs rest for label_1:
ACC of 0th validation : 0.500
ACC of 1th validation : 0.667
ACC of 2th validation : 0.667
ACC of 3th validation : 0.600
ACC of 4th validation : 0.667
one vs rest for label_2:
ACC of 0th validation : 0.833
ACC of 1th validation : 0.700
ACC of 2th validation : 0.900
ACC of 3th validation : 0.767
ACC of 4th validation : 0.933

Out[119]: 0.8111111111111111
```

Logistic Regression 在 Iris 数据集上ovr 5 折交叉验证结果如下

```
In  [145]: Cross_validation(samples, labels, multiclass=True, k=5)

one vs rest for label_0:
ACC of 0th validation : 1.000
ACC of 1th validation : 1.000
ACC of 2th validation : 1.000
ACC of 3th validation : 1.000
ACC of 4th validation : 1.000
one vs rest for label_1:
ACC of 0th validation : 0.800
ACC of 1th validation : 0.567
ACC of 2th validation : 0.633
ACC of 3th validation : 0.600
ACC of 4th validation : 0.700
one vs rest for label_2:
ACC of 0th validation : 1.000
ACC of 1th validation : 0.900
ACC of 2th validation : 0.933
ACC of 3th validation : 1.000
ACC of 4th validation : 0.967

Out[145]: 0.8733333333333333
```

根据结果得到每一个标签ovr的$confusion_matrix$，发现在第二类对另外两类的时候效果较差

```
one vs rest for label_0:
[[23  0]
 [ 0  7]]
[[21  0]
 [ 0  9]]
[[17  0]
 [ 0 13]]
[[22  0]
 [ 0  8]]
[[17  0]
 [ 0 13]]
```

```
one vs rest for label_1:
[[18  2]
 [ 4  6]]
[[17  1]
 [12  0]]
[[19  0]
 [11  0]]
[[15  5]
 [ 7  3]]
[[21  2]
 [ 7  0]]
```

```
one vs rest for label_2:
[[17  0]
 [ 0 13]]
[[18  3]
 [ 0  9]]
[[23  1]
 [ 1  5]]
[[18  0]
 [ 0 12]]
[[19  1]
 [ 0 10]]
```

SVM 在 Iris 数据集上分别使用$linear, rbf, poly, sigmoid$四种不同核函数进行 5 折交叉验证的结果如下

```
In [298]: for kernel in ['linear', 'rbf', 'poly', 'sigmoid']:
              print('kernel: %s' % (kernel))
              Cross_validation(samples, labels, k=5, kernel=kernel)
```

```
kernel: linear
ACC of 0th validation : 1.000
ACC of 1th validation : 0.900
ACC of 2th validation : 1.000
ACC of 3th validation : 1.000
ACC of 4th validation : 0.967
total acc: 0.973
kernel: rbf
ACC of 0th validation : 0.967
ACC of 1th validation : 0.900
ACC of 2th validation : 1.000
ACC of 3th validation : 1.000
ACC of 4th validation : 0.800
total acc: 0.933
kernel: poly
ACC of 0th validation : 1.000
ACC of 1th validation : 0.933
ACC of 2th validation : 1.000
ACC of 3th validation : 0.967
ACC of 4th validation : 0.933
total acc: 0.967
kernel: sigmoid
ACC of 0th validation : 0.233
ACC of 1th validation : 0.133
ACC of 2th validation : 0.133
ACC of 3th validation : 0.300
ACC of 4th validation : 0.200
total acc: 0.200
```

Naïve Bayes 在 Iris 数据集上进行 5 折交叉验证的结果如下

```
In [328]: Cross_validation(samples, labels, k=5)

          ACC of 0th validation : 0.967
          ACC of 1th validation : 0.900
          ACC of 2th validation : 1.000
          ACC of 3th validation : 1.000
          ACC of 4th validation : 0.833

Out[328]: 0.94
```

在 Iris 数据集上，Naïve Bayes 以及 SVM 的$linear, rbf, poly$三种 kernel 效果较好。线判别分析和对数几率回归只能进行二分类，需要用ovr的方法将多分类转化为二分类问题。

西瓜数据集 3.0

LDA 在西瓜 3.0 数据集上的 5 折交叉验证结果

```
In [43]: Cross_validation(samples, labels, multiclass=False, k=5)

         ACC of 0th validation : 0.667
         ACC of 1th validation : 0.667
         ACC of 2th validation : 0.667
         ACC of 3th validation : 0.667
         ACC of 4th validation : 1.000

Out[43]: 0.7333333333333333
```

Logistic Regression 在西瓜 3.0 数据集上的 5 折交叉验证结果

```
In [72]: Cross_validation(samples, labels, multiclass=False, k=5)

         ACC of 0th validation : 0.667
         ACC of 1th validation : 1.000
         ACC of 2th validation : 0.333
         ACC of 3th validation : 0.667
         ACC of 4th validation : 0.667

Out[72]: 0.6666666666666666
```

Naïve Bayes 在西瓜 3.0 数据集上的 5 折交叉验证结果

In [76]:
```
Cross_validation(samples, labels, flag, k=5)
```

```
ACC of 0th validation : 0.333
ACC of 1th validation : 0.333
ACC of 2th validation : 0.667
ACC of 3th validation : 0.667
ACC of 4th validation : 0.333
```

Out[76]:
```
0.4666666666666667
```

SVM 在 Iris 数据集上分别使用$linear, rbf, poly, sigmoid$四种不同核函数进行 5 折交叉验证的结果如下

In [79]:
```
for kernel in ['linear', 'rbf', 'poly', 'sigmoid']:
    print('kernel: %s' % (kernel))
    Cross_validation(samples, labels, k=5, kernel=kernel)
```

```
kernel: linear
ACC of 0th validation : 0.667
ACC of 1th validation : 0.667
ACC of 2th validation : 1.000
ACC of 3th validation : 0.667
ACC of 4th validation : 0.333
total acc: 0.667
kernel: rbf
ACC of 0th validation : 0.667
ACC of 1th validation : 0.667
ACC of 2th validation : 0.333
ACC of 3th validation : 0.333
ACC of 4th validation : 0.333
total acc: 0.467
kernel: poly
ACC of 0th validation : 0.667
ACC of 1th validation : 1.000
ACC of 2th validation : 1.000
ACC of 3th validation : 0.667
ACC of 4th validation : 0.333
total acc: 0.733
kernel: sigmoid
ACC of 0th validation : 0.333
ACC of 1th validation : 0.000
ACC of 2th validation : 0.333
ACC of 3th validation : 0.333
ACC of 4th validation : 0.000
total acc: 0.200
```

西瓜 3.0 数据集数量少，shuffle 之后多次测试精度波动较大。与 Iris 比较可以发现大量的数据对于机器学习的重要意义。