In [1]:

```
import scipy.io as scio
import numpy as np
from functools import reduce
import operator
```

读数据集函数

In [2]:

```
"""
xfile: x文件
yfile: y文件
rate: 训练集数量/测试集数量
"""
def readData(xfile, yfile, rate):
    # 加载文件， 此处是dict
    x = scio.loadmat(xfile)
    y = scio.loadmat(yfile)

    x = x.get("mnist_train")
    y = y.get("mnist_train_labels")

    # 二值化
    x = np.where(x > 0, 1, 0)

    num = x.shape[0]
    index = int(num * (rate / (rate + 1)))

    x_train = x[0:index]
    y_train = y[0:index]
    x_test = x[index:num]
    y_test = y[index:num]

    return x_train, y_train, x_test, y_test
```

读取训练集和测试集

In [3]:

```
x_train, y_train, x_test, y_test = readData("dataset/mnist_train.mat", "dataset/mnist_train_lab
print("x_train.shape = {}".format(x_train.shape))
print("y_train.shape = {}".format(y_train.shape))
print("x_test.shape = {}".format(x_test.shape))
print("y_test.shape = {}".format(y_test.shape))
```

```
x_train.shape = (50000, 784)
y_train.shape = (50000, 1)
x_test.shape = (10000, 784)
y_test.shape = (10000, 1)
```

计算流程p(y|x) = p(y)*p(x|y)

1. 计算p(y)
2. 计算p(x|y)
3. 计算p(y|x)

1.计算p(y)

In [4]:

```
# 按y把数据分成10组
# 初始化px_group
px_group = []
for i in range(10):
    px_group.append(i)
    px_group[i] = []

# px_group的第一维表示类别[0-9], 第二维表示相应的所有x样例
for i, y in enumerate(y_train):
    px_group[int(y)].append(x_train[i])

py = []
for i in range(10):
    py.append(len(px_group[i]) / x_train.shape[0])

py = np.array(py)
py = py.reshape(-1, 1)
print("py.shape = {}".format(py.shape))
print("py       = {}".format(py.squeeze()))
```

```
py.shape = (10, 1)
py       = [ 0.09864  0.11356  0.09936  0.10202  0.09718  0.09012  0.09902  0.1035
  0.09684  0.09976]
```

2.计算p(x|y)

In [5]:

```
pxy = []
for i in range(10):
    group = np.array(px_group[i])
    # 按列求和
    group = np.sum(group, axis=0) / len(px_group[i])
    pxy.append(group)

pxy = np.array(pxy)
print("pxy.shape = {}".format(pxy.shape))
```

```
pxy.shape = (10, 784)
```

3.计算p(yi|x)

(1) 多项式朴素贝叶斯

In [6]:

```
def predict_mul(x):
    result = []
    for i in range(10):
        # 把0换成1
        temp = x * pxy[i]
        resulti = []
        for j in np.where(temp == 0, 1, temp):
            py_pxy = py[i][0] * reduce(operator.mul, j)
            resulti.append(py_pxy)
        result.append(resulti)

    result = np.argmax(result, axis=0)
    return result
```

(2) 伯努力朴素贝叶斯

In [7]:

```
def predict_ber(x):

    result = []

    for i in range(10):
        # 把0换成1
        temp = x * pxy[i]
        resulti = []
        for j in np.where(temp == 0, 1-pxy[i], temp):
            py_pxy = py[i][0] * reduce(operator.mul, j)
            resulti.append(py_pxy)
        result.append(resulti)

    result = np.argmax(result, axis=0)
    return result
```

In [8]:

```
# 多项式朴素贝叶斯
result_mul = predict_mul(x_test)
# 伯努力朴素贝叶斯
result_ber = predict_ber(x_test)
```

In [9]:

```
acc_mul = np.where(result_mul == np.squeeze(y_test), 1, 0)
print("多项式朴素贝叶斯 Acc_mul = {}".format(np.sum(acc_mul) / len(result_mul)))

acc_ber = np.where(result_ber == np.squeeze(y_test), 1, 0)
print("伯努力朴素贝叶斯 Acc_ber = {}".format(np.sum(acc_ber) / len(result_ber)))
```

```
多项式朴素贝叶斯 Acc_mul = 0.626
伯努力朴素贝叶斯 Acc_ber = 0.8469
```