HistogramClassifier

April 28, 2015

THE CODE TO GENERATE DATA AND TO RUN THE CLASSIFIER

In [112]:
```
from __future__ import division
import pandas as pd
import numpy as np
import sklearn as sk

data = pd.read_csv('StudentLoanSimulatedData.csv')

# vars[0] is % of ParentsIncome dedicated to all of their children's educations
# vars[1] is # of hours worked by student per year
# vars[2] is education baseline (must be > 0)
# vars[3] is impact of grades on scholarship
# vars[4] is the negative impact of ParentsIncome on scholarship eligibility
vars = np.array([.15, 2000, 1, .25, .1])
N = 1000

def studentLoanData(vars,N):

    randMat = np.array(np.random.uniform(0,1,[N,20]))
    ParentsIncome = randMat[:,0]*100000
    NumberOfSiblings = np.floor(randMat[:,1]*10)
    ParentsPortion = np.divide(ParentsIncome*vars[0],1+NumberOfSiblings)

    WorkExperience = np.floor(randMat[:,2]*8)
    Education = np.floor(randMat[:,3]*2)
    Skills = np.floor(randMat[:,4]*4)
    MinimumWage = np.floor(randMat[:,5]*20)
    StudentIncome = vars[1]*MinimumWage+np.multiply(WorkExperience,Education+vars[2],Skills)

    Years = np.floor(randMat[:,6]*8)
    College = randMat[:,7]*10000
    InState = np.floor(randMat[:,8]*2)
    Tuition = np.multiply(Years,College,InState+1)

    Grades =  np.floor(randMat[:,9]*40)/10
    Scholarship = np.multiply(vars[3]*Grades,Tuition - vars[4]*ParentsIncome)
    Scholarship = Scholarship.clip(0)

    State = randMat[:,10]*15000
    Roomies = np.floor(randMat[:,11]*10)
    BlingFactor = np.floor(randMat[:,12]*9)
    LivingExpenses = np.divide(np.multiply(State,1+BlingFactor),1+Roomies)
```

```
    Materials = randMat[:,13]*5000

    StudentLoanAmount = ParentsPortion + StudentIncome + Scholarship - Tuition - LivingExpens
    Loans = (StudentLoanAmount<0)
    Loans = Loans.astype(int)

    NumberOfSiblings
    StudentIncome

    data = pd.DataFrame.from_items([('Loans', Loans), ('ParentsIncome', ParentsIncome),
                                    ('WorkExperience', WorkExperience), ('Education', Educati
                                    ('MinimumWage', MinimumWage), ('Grades', Grades),
                                    ('Tuition', Tuition), ('Years', Years), ('College', Colle
                                    ('InState', InState), ('BlingFactor', BlingFactor),
                                    ('NumberOfSiblings', NumberOfSiblings), ('State', State),

    return data

def histogramClass(data, testratio = .2, maxgrid = 2):
    if (str(type(data)) == "<class 'pandas.core.frame.DataFrame'>"):
        nrow= np.shape(data)[0]
        ncol = np.shape(data)[1]
        stop = int(np.ceil(nrow*(1-testratio)))
        testrow = int(nrow - stop)
        classifier = pd.DataFrame()
        bins = np.arange(0,1, 1/maxgrid)

        ##### Create grid lines
        for i in range(ncol):
            colmin = min(data.ix[:,i])
            colmax = max(data.ix[:,i])
            data.ix[:,i] = (data.ix[:,i]-colmin)/colmax

            if i>0:
                data.ix[:,i] = np.digitize(data.ix[:,i],bins)
            else:
                data.ix[:,i] = data.ix[:,i]

        histogram = data.loc[range(stop)]
        histogram2 = data.loc[range(stop+1, nrow)]



        ##### Store rows of histogram that predict "output = 1"
        vector = histogram.groupby(list(histogram.columns)[1:]).mean() >= .5
        a = vector.index.unique()
        classifier = dict()
        for i in a:
            classifier[i] = vector.ix[i]['Loans']

        count = 0
        results = np.zeros((testrow,1))

        for i in range(stop+1,nrow):
```

```
                    if classifier.has_key(tuple(histogram2.loc[i][1:])):
                    #if tuple(histogram2.ix[i,1:]) in classifier.keys():
                        results[i-(stop+1)] = int(classifier[tuple(histogram2.loc[i][1:])])
                    else:
                        results[i-(stop+1)] = 0
                    if results[i-(stop+1)] == histogram2.loc[i][0]:
                        count = count + 1

                accuracy = (count/testrow)
                return accuracy

            else:
                print "Incorrect Data Type - Data should be in Pandas DataFrame."
```

TRIAL WITH TWO BINS The classifiers converge to an accuracy that seems to be close to 80% as the number of observations goes to infinity.

```
In [128]: for i in [10,100,1000,10000,100000,1000000]:
              data = studentLoanData(vars,i)
              acc = histogramClass(data)
              print str(i).ljust(10) + str(acc).ljust(10) + '\n'


0.5
10        0.5

0.3
100       0.3

0.53
1000      0.53

0.688
10000     0.688

0.78595
100000    0.78595

0.7985
1000000   0.7985
```

TRIAL WITH THREE BINS The classifiers converge slowly at first, but then quickly increase in accuracy as the number of observations increases.

```
In [127]: for i in [10,100,1000,10000,100000,1000000]:
              data = studentLoanData(vars,i)
              acc = histogramClass(data, maxgrid = 3)
              print str(i).ljust(10) + str(acc).ljust(10) + '\n'


0.0
10        0.0

0.45
100       0.45
```

```
0.495
1000      0.495

0.492
10000     0.492

0.56455
100000    0.56455

0.771705
1000000   0.771705
```