MASSEY UNIVERSITY
Institute of Fundamental Sciences / School of Engineering and Advanced Technology

161.326 Statistical Machine Learning

Solutions to Assignment 1, 2010

Question 1

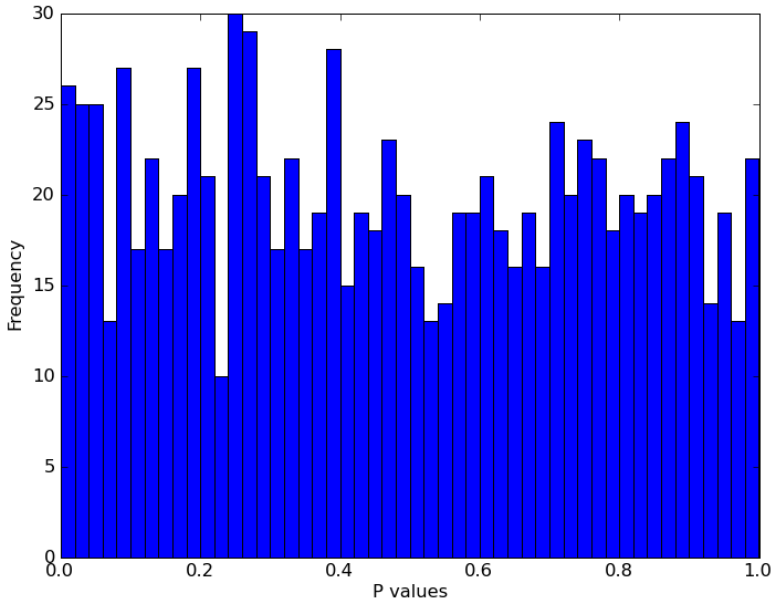
CODE

```
from pylab import *
from numpy import *
from scipy import stats  # necessary to obtain P-values

def linreg1(x,y):
    # takes Nx1 matrices x and y, and performs linear regression y = beta x
    x1 = concatenate((ones((shape(x)[0],1)),x),axis=1)
    V = linalg.inv(transpose(x1)*x1)
    betahat = V*transpose(x1)*y
    yhat = x1*betahat
    RSS = transpose(y - yhat)*(y - yhat)
    return betahat,yhat,RSS,V

def get_slope_p(x,y):
    # takes Nx1 matrices x and y, and performs linear regression y = beta x,
    # returning the P-value for the test of slope = 0
    N = len(x)
    temp = linreg1(x,y) # this avoids calling the function 3 times
    betahat = temp[0] # slope and intercept
    betahat1 = betahat[1] # slope
    RSS = temp[2]
    s2 = RSS/(N-2)
    V = temp[3]
    v1 = diag(V)[1]
    t1 = betahat1/sqrt(s2*v1) # t-vale for the slope = 0 test
    if t1 < 0:
        pvalue = 2*stats.t.cdf(t1, N-2)
    else:
        pvalue = 2*(1 - stats.t.cdf(t1, N-2)) # 2-sided test, remember
    return pvalue

numsamples = 1000
pvalues = zeros(numsamples)
pvalues2 = zeros(numsamples)
samplesize = 100
for i in range(numsamples):
    x = arange(samplesize)+1
    x2 = x
    x2[samplesize-1] = 10000
    x = transpose(matrix(x))
    x2 = transpose(matrix(x2))
    y = random.normal(loc=0.0, scale=1.0, size=size(x))
    y = transpose(matrix(y))
    pvalues[i] = get_slope_p(x,y)
    pvalues2[i] = get_slope_p(x2,y) # thus the two experiments differ only in
one x-value
```

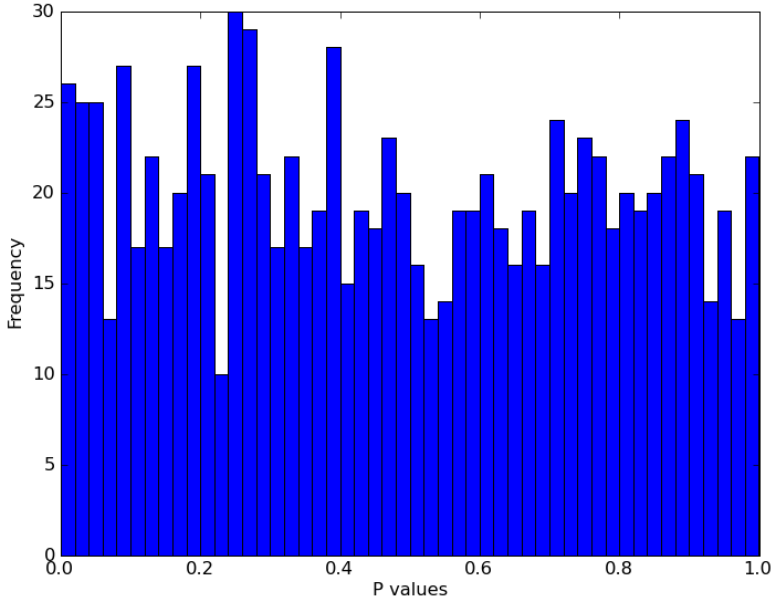
```
figure(1)
hist(array(pvalues),bins=50)    # histogram
xlabel('P values')
ylabel('Frequency')
figure(2)
hist(array(pvalues2),bins=50)    # histogram
xlabel('P values')
ylabel('Frequency')
show()
```

OUTPUT

for x:



for x2:



DISCUSSION

The P values are of course uniformly distributed under the null hypothesis. Recall that a significance level of α means that you will get $100\alpha\%$ 'false positives'. This is a consequence of having to set a threshold *somewhere*, as the t distribution has infinite support or, in other words, anything is *possible*. Note that the two histograms are identical (I hope you remembered to use the same y-values for each?)

COMMENT

Note that for a 2-sided test, which this is, the P-value is not just the CDF. See the last few lines of `get_slope_p()`

Question 4

CODE

```
from pylab import *
from numpy import *

def linregp_ni(X,y):
    # takes N x p matrix X and a N x 1 vector y, and fits
    # linear regression y = X*beta. NOTE indenting
    betahat = linalg.inv(transpose(X)*X)*transpose(X)*y
    yhat = X*betahat
    RSS = transpose(y - yhat)*(y - yhat)
    return betahat,yhat,RSS

def bin2dec(b): # converts a binary sequence into decimal
    d = 0
    for i in range(b.shape[1]):
        d = 2*d+b[0,i]
    return d

def dec2bin(d): # converts a decimal number into binary
    temp = d - 2*floor(0.5*d)
    if temp == 0:
        b = zeros((1,1))
    else:
        b = ones((1,1))
    d = 0.5*(d-temp)
    while d > 0:
        temp = d - 2*floor(0.5*d)
        b = concatenate((b,temp*ones((1,1))),axis=1)
        d = 0.5*(d-temp)
    ## pad out remaining indicies
    b = concatenate((b,zeros((1,9-b.shape[1]))),axis=1)
    return b

## training data
traindata = loadtxt('prostate_train.txt')
traindata = traindata.reshape(-1,9)
## reformat data as X and Y
ytrain = transpose(matrix(traindata[:,8]))
Xtrain= matrix(traindata[:,0:8])
## standardize
```

```
covXtrain = cov(transpose(matrix(Xtrain)))
sdXtrain = sqrt(diag(covXtrain))
for i in range(8):
  Xtrain[:,i] = Xtrain[:,i]/sdXtrain[i]

## test data
testdata = loadtxt('prostate_test.txt')
testdata = testdata.reshape(-1,9)
## reformat data as X and Y
ytest = transpose(matrix(testdata[:,8]))
Xtest = matrix(testdata[:,0:8])
## standardize same as training data
for i in range(8):
  Xtest[:,i] = Xtest[:,i]/sdXtrain[i]

results = zeros((2**9 - 1,12)) # each row has:
                               # modelid (included predictors in decimal)
                               # number of predictors
                               # RSS
                               # beta (last 9 columns)
modelno = -1
for j0 in range(2):
    for j1 in range(2):
        for j2 in range(2):
            for j3 in range(2):
                for j4 in range(2):
                    for j5 in range(2):
                        for j6 in range(2):
                            for j7 in range(2):
                                for j8 in range(2):
                                    X = matrix(zeros(shape(ytrain)))  # dummy
column
                                    if j0:
                                        X =
concatenate((X,ones(shape(ytrain))),axis=1)
                                    if j1:
                                        X = concatenate((X,Xtrain[:,0]),axis=1)
                                    if j2:
                                        X = concatenate((X,Xtrain[:,1]),axis=1)
                                    if j3:
                                        X = concatenate((X,Xtrain[:,2]),axis=1)
                                    if j4:
                                        X = concatenate((X,Xtrain[:,3]),axis=1)
                                    if j5:
                                        X = concatenate((X,Xtrain[:,4]),axis=1)
                                    if j6:
                                        X = concatenate((X,Xtrain[:,5]),axis=1)
                                    if j7:
                                        X = concatenate((X,Xtrain[:,6]),axis=1)
                                    if j8:
                                        X = concatenate((X,Xtrain[:,7]),axis=1)
                                    if X.shape[1]>1: # at least 1 predictor
                                        modelno = modelno+1
                                        modelid =
bin2dec(matrix([[j8,j7,j6,j5,j4,j3,j2,j1,j0]]))
                                        #print(dec2bin(modelid))
                                        results[modelno,0] = modelid # included
predictors
                                        numpred = j0+j1+j2+j3+j4+j5+j6+j7+j8 #
number of predictors
                                        results[modelno,1] = numpred
```

```
                                        if modelid > 1: # not just intercept
                                            X = X[:,1:X.shape[1]+1] # remove
dummy column
                                            fit = linregp_ni(X,ytrain)
                                            results[modelno,2] = fit[2] # RSS
                                            ## put beta in correct columns:
                                            index = dec2bin(modelid)
                                            j = 0
                                            for i in range(9):
                                                if index[0,i] > 0.5:
                                                    results[modelno,3+i] =
fit[0][j]
                                                    j = j+1
                                        else:  # just intercept
                                            yhat =
ones(shape(ytrain))*mean(ytrain)
                                            results[modelno,2] =
transpose(ytrain - yhat)*(ytrain - yhat)
                                            results[modelno,3] = mean(ytrain)
#print results[results.shape[0]-1,:]
bestresults = zeros((9,12))
print 'best models on training data'
for i in range(9):
    bestresults[i] = i+1 # number of predictors
    bestRSS = max(results[:,2])
    for j in range(results.shape[0]):
        if results[j,1] == i+1:
            #print results[j,0:3]
            if results[j,2] < bestRSS:
                bestpred = results[j,0]
                bestRSS = results[j,2]
                bestbeta = results[j,3:12]
    bestresults[i,1] = bestpred
    bestresults[i,2] = bestRSS
    bestresults[i,3:12] = bestbeta
    print 'numpred=',i+1,', RSS=',bestRSS,', predictors=',dec2bin(bestpred)

#print bestresults[:,0:3]

Xtest = concatenate((ones(shape(ytest)),Xtest),axis = 1)
print 'best models on test data'
for i in range(9):
    beta = bestresults[i,3:12]
    ytesthat = Xtest*transpose(matrix(beta))
    RSS = float(transpose(ytest - ytesthat)*(ytest - ytesthat))
    print 'numpred=',i+1,', RSS=',RSS,', predictors=',dec2bin(bestresults[i,1])
```

OUTPUT

```
best models on training data
numpred= 1 , RSS= 78.1269427781 , predictors= [[ 0. 0. 1. 0. 0. 0. 0. 0. 0.]]
numpred= 2 , RSS= 38.2927785037 , predictors= [[ 0. 1. 1. 0. 0. 0. 0. 0. 0.]]
numpred= 3 , RSS= 36.0480287023 , predictors= [[ 0. 1. 1. 0. 0. 1. 0. 0. 0.]]
numpred= 4 , RSS= 32.9074073606 , predictors= [[ 0. 1. 1. 0. 1. 1. 0. 0. 0.]]
numpred= 5 , RSS= 32.2540546588 , predictors= [[ 0. 1. 1. 0. 1. 1. 0. 0. 1.]]
numpred= 6 , RSS= 30.9750174411 , predictors= [[ 0. 1. 1. 0. 1. 1. 1. 0. 1.]]
numpred= 7 , RSS= 29.4691574688 , predictors= [[ 0. 1. 1. 1. 1. 1. 1. 0. 1.]]
numpred= 8 , RSS= 29.437300581 , predictors= [[ 1. 1. 1. 1. 1. 1. 1. 0. 1.]]
numpred= 9 , RSS= 29.4263839583 , predictors= [[ 1. 1. 1. 1. 1. 1. 1. 1. 1.]]
```

```
best models on test data
numpred= 1 , RSS= 34.2617309292 , predictors= [[ 0. 0. 1. 0. 0. 0. 0. 0. 0.]]
numpred= 2 , RSS= 15.0114714546 , predictors= [[ 0. 1. 1. 0. 0. 0. 0. 0. 0.]]
numpred= 3 , RSS= 12.2496906867 , predictors= [[ 0. 1. 1. 0. 0. 1. 0. 0. 0.]]
numpred= 4 , RSS= 14.907203104 , predictors= [[ 0. 1. 1. 0. 1. 1. 0. 0. 0.]]
numpred= 5 , RSS= 15.5678986166 , predictors= [[ 0. 1. 1. 0. 1. 1. 0. 0. 1.]]
numpred= 6 , RSS= 16.8402851759 , predictors= [[ 0. 1. 1. 0. 1. 1. 1. 0. 1.]]
numpred= 7 , RSS= 17.8186825423 , predictors= [[ 0. 1. 1. 1. 1. 1. 1. 0. 1.]]
numpred= 8 , RSS= 17.4608187118 , predictors= [[ 1. 1. 1. 1. 1. 1. 1. 0. 1.]]
numpred= 9 , RSS= 17.5898669677 , predictors= [[ 1. 1. 1. 1. 1. 1. 1. 1. 1.]]
```

DISCUSSION

So the best model on the test data is a 3 predictor model using predictors 1,2 and 5. Note that the intercept term is not included. While extra predictors always improve the fit on the data they are optimised to, they can result in over fitting against independent data.

COMMENTS

The 9 nested for loops can of course be replaced by one loop over 1 to 511, and conversion of the loop number into a binary representation of the model.