Part-1 of the homework is separate, turned in on paper.
Shunyi print out:

```
Shunyi Data -- part1 -- using only python standard libraries
Using X input labels:
['month', 'hour', 'TEMP', 'PRES', 'DEWP', 'RAIN', 'WSPM']
Using Y ouput labels:
['PM2.5', 'PM10', 'SO2', 'NO2', 'CO', 'O3']
Data contains  30194  valid entries
initial B matrix:
[1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1]
learning paramter alpha  3e-12
Stopping criteria defined by dB < max( |Bnew[i,j] - Bold[i,j]| )
Stopping point set to: .001
1 iteration dB =  0.189846527927
2 iteration dB =  0.154492322757
3 iteration dB =  0.125721924316
4 iteration dB =  0.102309268806
5 iteration dB =  0.0832566151197
6 iteration dB =  0.0677520256328
7 iteration dB =  0.0551347651149
8 iteration dB =  0.0448671431902
9 iteration dB =  0.0365116004146
10 iteration dB =  0.0297120614787
11 iteration dB =  0.024178760894
12 iteration dB =  0.0196758945028
13 iteration dB =  0.0160115705761
14 iteration dB =  0.0130296322581
15 iteration dB =  0.0106030028706
16 iteration dB =  0.00862827048149
17 iteration dB =  0.00702128095742
18 iteration dB =  0.00571355169687
19 iteration dB =  0.00464935321237
20 iteration dB =  0.00378333419276
21 iteration dB =  0.00307858883615
22 iteration dB =  0.00250508409211
23 iteration dB =  0.00203837978947
24 iteration dB =  0.00165858710706
25 iteration dB =  0.0013495210025
26 iteration dB =  0.00109801047912
27 iteration dB =  0.000893337297883
final Bnew:
[0.9937476824803185, 0.9938573176843345, 0.9932887939628414, 0.9935586676248743, 1.001708704004432, 0.9935206636854815]
[0.988719267662102, 0.9890771672707569, 0.9879746396041101, 0.9883702960970319, 1.000843291550198, 0.9889279003211554]
[0.9862662087561694, 0.986657278932259, 0.9855681031673413, 0.9857836522827714, 0.9822493854317195, 0.9882559757275126]
[0.048530857197721856, 0.06812186962076881, -0.015520415003358773, 0.01463465461936806, 1.15285839556411, 0.024605307556908436]
[0.9978776555671177, 0.9978284988481917, 0.9966456296523946, 0.996900308993478, 0.9979641765939774, 0.9981699284951762]
[0.9999371193154755, 0.9999338985927453, 0.9999335489602103, 0.9999323345245065, 0.999970566070429, 0.9999437016425048]
[0.9981327414713921, 0.9981933133966185, 0.9981424780228455, 0.9981356626707909, 0.9980741213513982, 0.9983412712467779]

start numpy checks to verify output
initial B matrix
[[1 1 1 1 1 1]
 [1 1 1 1 1 1]
 [1 1 1 1 1 1]
 [1 1 1 1 1 1]
 [1 1 1 1 1 1]
 [1 1 1 1 1 1]
 [1 1 1 1 1 1]]
2 iteration dB =  0.1898465279265339
3 iteration dB =  0.15449232275714453
4 iteration dB =  0.1257219243155091
5 iteration dB =  0.10230926880640995
6 iteration dB =  0.08325661511968119
7 iteration dB =  0.06775202563282895
8 iteration dB =  0.055134765114935785
9 iteration dB =  0.044867143190188974
10 iteration dB =  0.03651160041463311
11 iteration dB =  0.029712061478746568
12 iteration dB =  0.024178760893971155
13 iteration dB =  0.019675894502831914
14 iteration dB =  0.016011570576057857
15 iteration dB =  0.013029632258074964
16 iteration dB =  0.010603002870615549
```

```
17 iteration dB =  0.008628270481490192
18 iteration dB =  0.007021280957423333
19 iteration dB =  0.005713551696867729
20 iteration dB =  0.004649353212368231
21 iteration dB =  0.0037833341927619385
22 iteration dB =  0.0030785888361455323
23 iteration dB =  0.002505084092113136
24 iteration dB =  0.0020383797894676942
25 iteration dB =  0.0016585871070582264
26 iteration dB =  0.001349521002497472
27 iteration dB =  0.0010980104791190846
28 iteration dB =  0.0008933372978827056
final Bnew from numpy:
[[ 0.99374768  0.99385732  0.99328879  0.99355867  1.0017087   0.99352066]
 [ 0.98871927  0.98907717  0.98797464  0.9883703   1.00084329  0.9889279 ]
 [ 0.98626621  0.98665728  0.9855681   0.98578365  0.98224939  0.98825598]
 [ 0.04853086  0.06812187 -0.01552042  0.01463465  1.1528584   0.02460531]
 [ 0.99787766  0.9978285   0.99664563  0.99690031  0.99796418  0.99816993]
 [ 0.99993712  0.9999339   0.99993355  0.99993233  0.99997057  0.9999437 ]
 [ 0.99813274  0.99819331  0.99814248  0.99813566  0.99807412  0.99834127]]
```

Code:

```
import csv
def printmat(matrix):
for row in matrix:
print(row)
def column(matrix, i):
col = i[0]
return [row[col] for row in matrix]

def transpose(matrix):
return zip(*matrix)

def prodElement(arow,bcol):# this is row * a column summed
return sum(map(lambda x,y:x*y,arow,bcol))

def prodRow(A, B, bcol):#all rows of a times specified col of B #returns a row of a matrix (C)
return [ prodElement(A[irow],column(B,[bcol])) for irow in range(len(A)) ]

def mult(A,B):
return transpose([ prodRow(A,B,bcol) for bcol in range(len(B[0])) ])

def scalar(alpha, A):
return  [list(map((alpha).__mul__, arow)) for arow in A]

def add(A, B):
return [[A[i][j] + B[i][j]  for j in range(len(A[0]))] for i in range(len(A))]

def sub(A, B):#A-B
return [[A[i][j] - B[i][j]  for j in range(len(A[0]))] for i in range(len(A))]

def maxDelta(Bold,Bnew):
deltaB = sub(Bold,Bnew)
deltaB = [[ abs(deltaB[i][j]) for j in range(len(deltaB[0])) ] for i in range(len(deltaB))]
#print "deltaB", deltaB
#print deltaB
#maxdB = [ max(x) for x in deltaB ]
#deltaB =map(abs,deltaB)
return max(map(max, deltaB))
#return max(maxdB)
#return


def Bp1(B,alpha,Xt,X,Cinv,Y):
#let Cinv b Identity
#XtC = mult(Xt,Cinv)
XtC = Xt
a2XtC = scalar(alpha*2,XtC)
XB = mult(X,B)
YmXB = sub(Y,XB)

return add(B, mult(a2XtC,YmXB ))


with open('PRSA_Data_Shunyi_20130301-20170228.csv') as csvfile:
list_1 = csv.reader(csvfile, delimiter=',', quotechar='|') #reads in raw data into matrix
list_2 = [x for x in list_1 if 'NA' not in x]  # deletes entries which contain NA
```

```
#create a subset value based matrix with labels extracted
#list_3 = list_2[1:]
list_3 = list_2[1:]

#print list_3B,alpha,Xt,X,Cinv,Y
#for row in list_3:
# print(’, ’.join(row))

# labels/vars of interest for output/input
xElements = ["month", "hour", "TEMP", "PRES", "DEWP", "RAIN", "WSPM"]
yElements = ["PM2.5", "PM10", "SO2", "NO2", "CO", "O3"]



#get data labels
labels = list_2[0]
#process labels (extract weird " ")
labels = [ x[1:-1] for x in labels ]
#print labels


#we extract columnwise -> each list in the 2d list is the respective elements of that variable
#this means columnwise will make x and y be transposed
#Xt, Yt=[],[]
Xt = [ column(list_3, [ilabel for ilabel in range(len(labels)) if labels[ilabel] == element] ) for element in xElements]
#x goes by nx(P+1) add a constant column
#const = [ 1. for x in Xt[0] ]
#Xt.append(const)
X = transpose(Xt)
#convert to fp
Xt = [ map(float, a) for a in Xt ]
X = [ map(float, a) for a in X ]
#print xElements
#print "xt "
#printmat(Xt)

#print "x "
#printmat(X)

Yt = [ column(list_3, [ilabel for ilabel in range(len(labels)) if labels[ilabel] == element] ) for element in yElements]
Y = transpose(Yt)
Y = [ map(float, a) for a in Y ]
#print yELements
#print Yt
#print "y "
#printmat(Y)

#print data information
print "Shunyi Data -- part1 -- using only python standard libraries"
print "Using X input labels:"
print xElements
print "Using Y ouput labels:"
print yElements
print "Data contains ", len(Y), " valid entries"


#create first B matrix intitialized to all 1s of proper dimension
# y(nxk) = X(nxp)B(pxk) + E(nxk)
B = [ len(Y[0])*[1] for colx in range(len(X[0])) ]
Bcopy = B
#B = []
#print "B "
print "initial B matrix: "
printmat(B)

alpha = 3e-12
print "learning paramter alpha ", alpha


""" testing
Bnew = Bp1(B,alpha,Xt,X,1,Y)
#printmat(Bnew)
Bnewnew = Bp1(Bnew,alpha,Xt,X,1,Y)
printmat(Bnewnew)
"""

Cinv = 1
```

```
dB = 1
it_cnt =1
Bnew = []
print "Stopping criteria defined by dB < max( |Bnew[i,j] - Bold[i,j]| )"
print "Stopping point set to: .001"
while dB > .001:
Bnew = Bp1(Bcopy,alpha,Xt,X,Cinv,Y)
dB = maxDelta(Bcopy, Bnew)
Bcopy = Bnew
print it_cnt, "iteration dB = ", dB
it_cnt = it_cnt+1
#if(it_cnt >= 6):
# break

print "final Bnew:"
printmat(Bnew)


print " "
print "start numpy checks to verify output"

import numpy
x1 = numpy.array(X)
y1 = numpy.array(Y)
B1 = numpy.array(B)
print "initial B matrix"
print B1
#print x1
#print y1
#print B1


"""
xb = numpy.dot(x1,B1)
ymxb = numpy.subtract(y1,xb)
Bnew1 = numpy.add(B1, numpy.dot(alpha*2.*x1.T, ymxb))
#print Bnew1
xb = numpy.dot(x1,Bnew1)
ymxb = numpy.subtract(y1,xb)
Bnew2 = numpy.add(Bnew1, numpy.dot(alpha*2.*x1.T,ymxb))
print Bnew2
"""
dB1 = 1
it_cnt1 = 1
while dB1 > .001:
xb = numpy.dot(x1,B1)
ymxb = numpy.subtract(y1,xb)
a2xt = (alpha*2.)*x1.T
Bnew1 = numpy.add(B1, numpy.dot(a2xt, ymxb) )
#dB1 = maxDelta(B1, Bnew1)
delta = numpy.subtract(B1,Bnew1)
delta = numpy.absolute(delta)
dB1 = numpy.amax(delta)
B1 = Bnew1
it_cnt1 = it_cnt1+1

print it_cnt1, "iteration dB = ", dB1

print "final Bnew from numpy:"
print Bnew1
```