Matrix Programming

Principal Component Analysis

Copyright © Software Carpentry 2010

This work is licensed under the Creative Commons Attribution License

See http://software-carpentry.org/license.html for more information.

So far, we have seen that NumPy is a numerical library in Python that provides access to linear algebra structures like matrices and vectors.

- It provides many tools for manipulating and accessing arrays.
- It provides a large library of linear algebraic operations.
- Numpy lets you focus on your problem area rather than on reinventing the (linear) wheel.

In this lecture, we'll take a look at a complete program that uses NumPy to compute the a principal components analysis (PCA) on a data set.

Don't worry about what PCA is, exactly. If you are interested, you can take a look at the Wikipedia entry. The important this is:

PCA is (just an) eigenvalue problem

The program is broken into three parts:

1) Read the data from a file

2) Prep data and compute eigenvalues (1 line!)

3) Make a skree plot, which is a visual depiction of the magnitude of the eigenvalues.

1) Read the data

```
#
# First import some things and get
# the file handle.
#
import sys
import numpy
import matplotlib.pyplot
# don’t worry about that last one ☺
arg = sys.argv[1]
fileHandle = open(arg, ’r’)
```

1) Read the data

The data is tab delimited, and it has a single label column that we pop off below.

```
text = [];
fileHandle.readline()
for line in fileHandle:
   lineArr = line.split(”\t”)
   lineArr.pop(0)
   text.append(map(float, lineArr))
```

2) Prep data ...

```
data = numpy.array(text)
numSamples = data.shape[1]

if(numSamples < 2):
  print ”Too few samples.  Need at least 2
  for well defined covariance matrix.\n”
  quit()
```

2) Prep data (build a covariance matrix)

```
rowMean = data.mean(axis=0)
rowSz = data.shape[0]
for i in range(rowSz):
  data[i,:] = data[i,:] - rowMean


covMatrix = numpy.mat(data).T *
  numpy.mat(data) # data’ * data
covMatrix = numpy.divide(covMatrix,
  numSamples-1)
```

2) Prep data ... and compute eigenvalues

```
v,w = numpy.linalg.eig(covMatrix)
```

That is why we want to use a tool like NumPy!

3) Plot the eigenvalues

matplotlib is a toolbox for making plots in Python. We won’t go in to details here, but I created a simple plot of the eigenvalues in order of magnitude.

```
matplotlib.pyplot.plot(v)

matplotlib.pyplot.show()
```

Summary:

Good software is modularized: a tool should do one thing and it should do it well.

NumPy is a tool that is designed to do linear algebra very well.

While many of the operations “feel like” list operations, be careful: arrays have some differences.

The key difference between lists and arrays is that arrays are built for speedy handling of large data sets of homogenous type.

created by

Richard T. Guy

November 2010

Copyright © Software Carpentry 2010

This work is licensed under the Creative Commons Attribution License

See http://software-carpentry.org/license.html for more information.