MATH 728D: Machine Learning Lab #11: Principal Component Analysis

John Burkardt

December 5, 2018

I have a huge number of data samples; can I find a small number of trends that explain them?

The second item of data we collect is probably quite different from the first. But unless our data is completely random, we expect that in a large sample, many date items are very similar. In fact, it's probably true that most of the data values are close to a small number of typical cases.

In principal component analysis (PCA), we try to recombine our data in ways that reveals the most information in the fewest number of components. We will review two methods for carrying out PCA, apply each to the same problem, and finish with an application involving image analysis.

1 Eigenvalues and Singular Values

There are two common approaches to PCA, both based on linear algebra. Assuming the data is stored as an $m \times n$ array A, we can essentially do an eigenvalue analysis of the symmetric matrix $A'A$, or we can apply the singular value decomposition (SVD) directly to A.

In this exercise, we will simply make sure you are familiar with the MATLAB commands needed, and check that the factorizations are correct.

Exercise 1:

- Create the matrix `A` by calling `rand(6,4)`;
- Compute `ATA` as the transpose of `A` times `A`;
- Use the `eig()` function to get `X`, the eigenvector and `L`, the eigenvalue matrices associated with `A`;
- For a single eigenvalue, the eigenvalue equation is written $Ax = \lambda x$;. Let `x` be the first column of `X`, and `lambda` be `L(1,1)`, and verify that `x` and `lambda` satisfy the eigenvalue equation for `ATA`;
- For all the eigenvalues of a square system, write $A = X'LX$; Verify that `X` and `L` satisfy this for `ATA`;
- Now compute `U, S, V` using the `svd()` function;
- Verify that these factors satisfy $A = USV'$;
- Use the first three SVD components to approximate `A`:

```
A3 = U(:,1:3) * S(1:3,1:3) * V(:,1:3)';
```

and compare this to `A`;

2 PCA by Eigenvalues of A'A

We will load a set of data about the chemical composition of various glass samples. We will ignore the first and last columns, copying the remaining data into an $m \times n$ array A. We "center" each column by subtracting off its mean. We create an $n \times n$ matrix by computing $ATA = A'A$. We compute the eigenvalues

and eigenvectors of ATA. The eigenvectors with largest eigenvalues indicate the components that have the most information in them. If we plot the eigenvalues, we can see the information content of each component.

Exercise 2:

- Use `csvread()` to read *glass_data.csv* into the variable `data`;
- Create the matrix A from entries `data(:,2:end-1)`;
- Get column means `mu=mean(A)` and subtract from each column;
- Compute `ATA` by multiplying the transpose of A times A;
- Get eigenvector and eigenvalue matrices, and extract the diagonal:

```
[ VEC, VAL ] = eig ( ATA );
lam = diag ( VAL );
```

- Plot the eigenvalues; you should see a sharp drop in magnitude;
- Plot the variance captured by the first K eigenvectors:

```
y = [ 0.0; cumsum(lam) ] / sum ( lam );
plot ( 0:9, y );
```

Based on your second plot, how many of the 9 eigenvectors would you need to use in order to account for at least 80% of the variance in the original data?

3 PCA by Singular Value Decomposition of A

PCA can also be carried out using the singular value decomposition. Given a choice, the SVD should be preferred. For one thing, it works directly with the matrix A rather than with $A'A$, so the condition number (resistance to error growth) is better. For the SVD algorithm, the importance of each singular vector is represented by the square of the corresponding singular value.

Exercise 3:

- Use `csvread()` to read *glass_data.csv* into the variable `data`;
- Create the matrix A from entries `data(:,2:end-1)`;
- Get column means `mu=mean(A)` and subtract from each column;
- Use the `svd()` command to get factors `U, S, V`, extract singular values from `S` and square them:

```
[ U, S, V ] = svd ( A );
svec = diag ( S );
svec = svec.^2;
```

- Plot the squared singular values; you should see a sharp drop in magnitude;
- Plot the variance captured by the first K singular vectors:

```
y = [ 0.0; cumsum(svec) ] / sum ( svec );
plot ( 0:9, y );
```

Your SVD plots should be almost identical to the eigenvalue plots.

4 PCA for Image Data

Consider an black and white image as an $m \times n$ array A of numbers. Now imagine each column of this array as being a data item, so that we have n items, each of dimension m. Note that usually, we have thought of the *rows* of an array as being the data items. However, for the SVD, it will be more natural to work with columns.

If we compute the SVD of A, then we will be able to approximate the original image using combinations of a small set of singular vectors. We also can compute automatically whether we have captured most of the information in the image. **Exercise 3:**

- Use the `imread()` function to read the image *casablanca.png* into array `I`, which is of type "unsigned integer 8 bit" or `uint8` ;•
- Set `m, n` to the dimensions of `I`;
- Use the `double()` function to create `A`, a real copy of `I`;
- Use the `svd()` function to get factors `U, S, V` of `A`;
- Extract the singular values, square them, and plot them:

```
svec = diag ( S );
svec = svec.^2;
plot ( 1:length(svec), svec, 'bo-' );
```

- Create a compressed image using K=10 components:

```
k = 10;
I2 = U(:,1:k) * S(1:k,1:k) * V(:,1:k)';
I2_min = min ( min ( I2 ) );
I2_max = max ( max ( I2 ) );
I2 = 255 * ( I2 - I2-min ) / ( I2_max - I2_min );
I2 = uint8 ( I2 );
imshow ( I2 );
```

- Repeat the previous step with K=20 components;