4 - Dimensionality Reduction

June 19, 2016

```
In [1]: %matplotlib notebook
```

```
/Users/AriaW/anaconda/lib/python2.7/site-packages/IPython/kernel/__init__.py:13: ShimWarning: The 'IPytho
  "You should import from ipykernel or jupyter_client instead.", ShimWarning)
```

```
In [2]: import numpy as np
        from dimensionality_reduction import generate_A, generate_B, generate_C
        from dimensionality_reduction import plot_3d, plot_pca, plot_lowdim, PCA
```

Dimensionality reduction is a general technique to find low-dimensional representations for high-dimensional stimuli. In this problem, we'll examine the classical approach called principal component analysis (PCA).

0.1 Part A (1 point)

Run each of the following cells to generate three datasets and then plot each dataset in a separate figure so you can see what they each look like. Note that the colors of the points in the figures have no meaning; they're only colored to help you visualize how the points are distributed in space. You can click and grab the images in order to rotate them and see the data from multiple angles. This will help you better visualize the points.

```
In [3]: # Generate dataset A and show the first few values
        A_data, A_colors = generate_A()
        A_data[:10]
```

```
Out[3]: array([[-0.98626109, -0.98626109],
               [ 0.8599553 ,  0.8599553 ],
               [ 0.22678625,  0.22678625],
               [-1.35911078, -1.35911078],
               [-0.5368619 , -0.5368619 ],
               [ 1.4396997 ,  1.4396997 ],
               [-2.1748806 , -2.1748806 ],
               [-0.40418839, -0.40418839],
               [ 1.09801697,  1.09801697],
               [-0.79225085, -0.79225085]])
```

```
In [4]: # Show dataset A
        plot_3d(A_data, A_colors, "Dataset A")
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
/Users/AriaW/anaconda/lib/python2.7/site-packages/matplotlib/collections.py:590: FutureWarning: elementw
  if self._edgecolors == str('face'):
```

```
In [5]: # Generate dataset A and show the first few values
        B_data, B_colors = generate_B()
        B_data[:10]
```

```
Out[5]: array([[-0.59300515,  0.83874629, -0.03188742],
               [-0.26103755, -1.42470877, -0.82049498],
               [ 0.10199788,  1.56713752,  0.77484747],
               [-2.321367  , -0.78684841, -1.95458283],
               [ 0.85902581, -0.63527786,  0.30680923],
               [-0.62020007, -0.17869599, -0.50803353],
               [ 1.14527485, -0.3738167 ,  0.62175155],
               [-0.53064481, -0.47794974, -0.58080446],
               [ 2.81329605,  2.53890684,  3.08146487],
               [ 1.07538241,  0.56450628,  0.99537641]])
```

```
In [6]: plot_3d(B_data, B_colors, "Dataset B")
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
In [7]: # Generate dataset C and show the first few values
        C_data, C_colors = generate_C()
        C_data[:10]
```

```
Out[7]: array([[  2.24239956e-01,  -1.39404787e+00,  -1.59969729e+00],
               [  2.33717612e-01,  -1.69268770e-02,   8.77541603e-01],
               [  6.29022775e-01,  -7.04077291e-01,   4.52354914e-01],
               [ -1.59196401e+00,   4.66236843e-01,  -7.19348164e-01],
               [  5.84576453e-01,  -6.06175610e-04,  -1.49573612e+00],
               [ -1.24723143e+00,  -1.54687653e+00,   7.47356196e-01],
               [  1.31184613e-01,  -6.04933319e-01,   1.87264372e+00],
               [  3.74532861e-02,  -3.68612506e-02,  -1.62358873e+00],
               [ -1.64144290e+00,   2.40281621e-01,   1.65627152e-01],
               [ -9.28178452e-01,  -1.31746000e+00,   9.42855780e-01]])
```

```
In [8]: plot_3d(C_data, C_colors, "Dataset C")
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

For each dataset, answer the following questions:

How many dimensions is the dataset embedded in? That is, how many dimensions does the data have as it is given?

How many dimensions does the dataset actually vary along? That is, what is the minimum number of dimensions you need to describe the data?

Dataset A is embedded in 2 dimensions but varies in 1 dimension. Dataset B is embedded in 3 dimensions, but varies in 2 dimensions. Dataset C is embedded in 3 dimensions but varies in 2 dimensions (it is like a rolled up sheet of paper, and a sheet of paper only has 2 dimensions).

0.2 Part B (1 point)

Run the following cells, which plots each dataset again, but additionally performs PCA on the data and plots some additional info.

```
In [9]: # Plot the A dataset
        plot_3d(A_data, A_colors, "A data")
        plot_pca(A_data, 1)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
In [10]: # Plot the B dataset
         plot_3d(B_data, B_colors, "B data")
         plot_pca(B_data, 2)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
In [11]: # Plot the C dataset
         plot_3d(C_data, C_colors, "C data")
         plot_pca(C_data, 2)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

Describe the figures produced by the previous three cells. What do the red and green vectors represent? What special property do the red and green vectors have with respect to each other, and why? It may help you to look at the code for `plot_pca`, `PCA` (which is called by `plot_pca`), and `np.linalg.eig` (which is called by `PCA`):

```
In [15]: # documentation for plot_pca
         plot_pca??
```

```
In [16]: # documentation for PCA
         PCA??
```

```
In [17]: # documentation for np.linalg.eig
         np.linalg.eig??
```

The red and green bars represent the principal components that capture the greatest variation in the dataset. They are actually the eigenvectors corresponding to the highest eigenvalues of the covariance matrix of the datasets. The special property the bars have with respect to each other is that they are orthogonal (or perpendicular to one another). This is because eigenvectors of a matrix have this special property.

0.3 Part C (1 point)

Run the following few cells, which plot each of the datasets in their low-dimensional coordinates found by the PCA algorithm.

```
In [18]: plot_lowdim(A_data, A_colors, "Dataset A")
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
In [19]: plot_lowdim(B_data, B_colors, "Dataset B")
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
In [20]: plot_lowdim(C_data, C_colors, "Dataset C")
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

How many dimensions are each of the datasets projected onto? For each dataset, explain whether or not PCA works well, and why.

PCA works well on dataset A since the dataset is originally just a line plotted in 2D. Dataset B can be projected into 2 dimensions and again PCA works well since the dataset is just a 2D plane originally plotted in 3D. Dataset C is projected into 2D but it does not look good. PCA doesn't work well on this dataset since it is nonlinear – the points lie along what's called a manifold. A more sophisticated dimensionality reduction technique would be required to project dataset C down to two dimensions in a way that makes sense.