customer_segments

February 26, 2016

1 Creating Customer Segments

In this project you, will analyze a dataset containing annual spending amounts for internal structure, to understand the variation in the different types of customers that a wholesale distributor interacts with.

Instructions:

- Run each code block below by pressing **Shift+Enter**, making sure to implement any steps marked with a TODO.
- Answer each question in the space provided by editing the blocks labeled "Answer:".
- When you are done, submit the completed notebook (.ipynb) with all code blocks executed, as well as a .pdf version (File > Download as).

```
In [3]: # Import libraries: NumPy, pandas, matplotlib
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from sklearn import preprocessing

        # Tell iPython to include plots inline in the notebook
        %matplotlib inline

        # Read dataset
        data = pd.read_csv("wholesale-customers.csv")
        print "Dataset has {} rows, {} columns".format(*data.shape)
        print data.head()  # print the first 5 rows
```

```
Dataset has 440 rows, 6 columns
   Fresh  Milk  Grocery  Frozen  Detergents_Paper  Delicatessen
0  12669  9656     7561     214              2674          1338
1   7057  9810     9568    1762              3293          1776
2   6353  8808     7684    2405              3516          7844
3  13265  1196     4221    6404               507          1788
4  22615  5410     7198    3915              1777          5185
```

```
/Users/paulreiners/anaconda/lib/python2.7/site-packages/matplotlib/font_manager.py:273: UserWarning: Mat
  warnings.warn('Matplotlib is building the font cache using fc-list. This may take a moment.')
```

1.1 Feature Transformation

1) In this section you will be using PCA and ICA to start to understand the structure of the data. Before doing any computations, what do you think will show up in your computations? List one or two ideas for what might show up as the first PCA dimensions, or what type of vectors will show up as ICA dimensions.

Answer: The features that maximize variance might show up as the first PCA dimensions. Linear combinations of the features will show up as ICA dimensions. These new features will maximize independence.

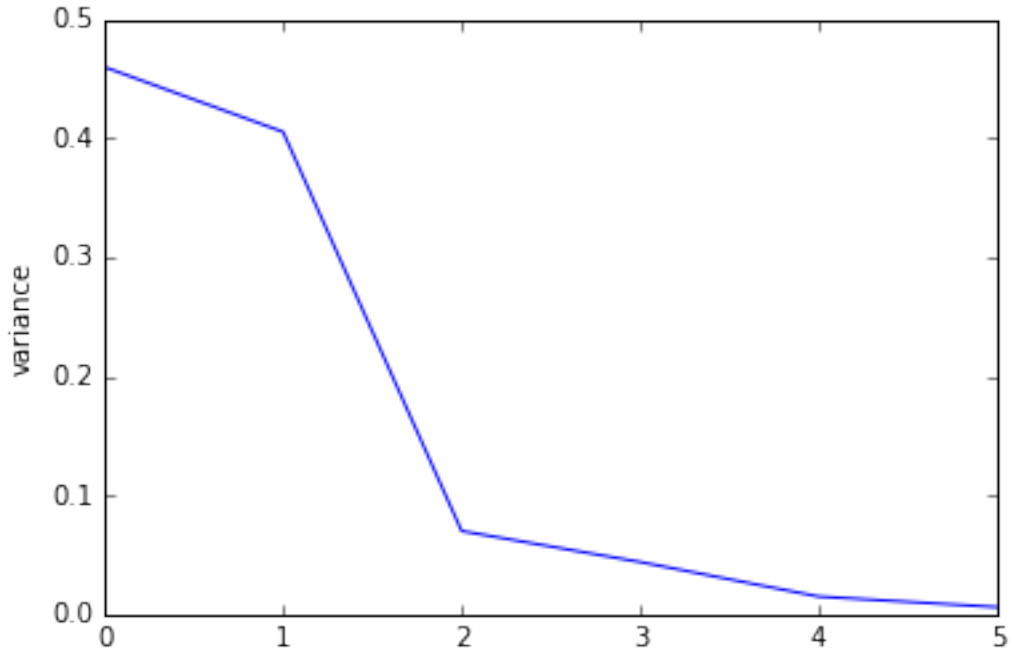
1.1.1 PCA

```
In [4]: # Apply PCA with the same number of dimensions as variables in the dataset
        from sklearn.decomposition import PCA
        pca = PCA(n_components=len(data.columns))
        pca.fit(data)

        # Print the components and the amount of variance in the data contained in each dimension
        print pca.components_
        print pca.explained_variance_ratio_
```

```
[[-0.97653685 -0.12118407 -0.06154039 -0.15236462  0.00705417 -0.06810471]
 [-0.11061386  0.51580216  0.76460638 -0.01872345  0.36535076  0.05707921]
 [-0.17855726  0.50988675 -0.27578088  0.71420037 -0.20440987  0.28321747]
 [-0.04187648 -0.64564047  0.37546049  0.64629232  0.14938013 -0.02039579]
 [ 0.015986    0.20323566 -0.1602915   0.22018612  0.20793016 -0.91707659]
 [-0.01576316  0.03349187  0.41093894 -0.01328898 -0.87128428 -0.26541687]]
[ 0.45961362  0.40517227  0.07003008  0.04402344  0.01502212  0.00613848]
```

2) How quickly does the variance drop off by dimension? If you were to use PCA on this dataset, how many dimensions would you choose for your analysis? Why?

```
In [5]: import matplotlib.pyplot as plt
        plt.plot(pca.explained_variance_ratio_)
        plt.ylabel('variance')
        plt.show()
```

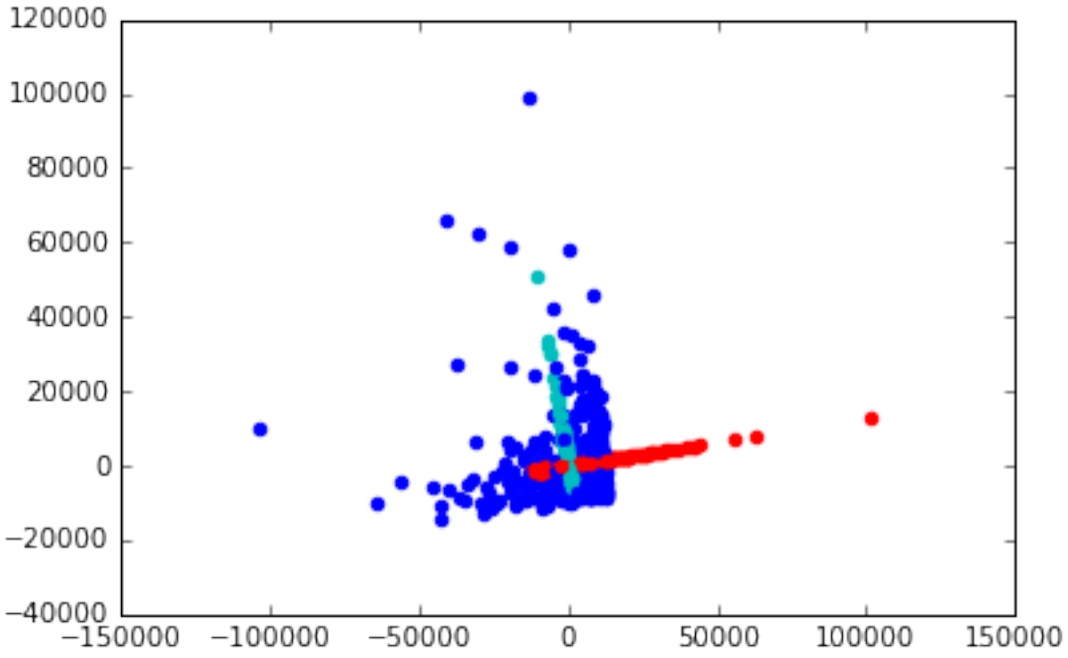


Answer: We can see how the variance drops off in the plot above. If i were to use PCA on this dataset, I would choose 3 dimensions for my dimensions. After 3 dimensions, the drop-off is less steep.

3) What do the dimensions seem to represent? How can you use this information?

```
In [22]: first_pc = pca.components_[0]
         second_pc = pca.components_[1]

         transformed_data = pca.transform(data)
         plt.close()
         for ii in transformed_data:
             plt.scatter(first_pc[0]*ii[0], first_pc[1]*ii[0], color="r")
             plt.scatter(second_pc[0]*ii[1], second_pc[1]*ii[1], color="c")
             plt.scatter(ii[0], ii[1], color="b")

         plt.show()
```

```
440
```



Answer: The dimensions seem to represent the features. We can use this information to find out which features are most useful in classifying data points. Looking at the first two components (in red and cyan), we can see that they are orthogonal.

1.1.2 ICA

```
In [8]: # Fit an ICA model to the data
        # Note: Adjust the data to have center at the origin first!
        data_scaled = preprocessing.scale(data)

        from sklearn.decomposition import FastICA
        ica = FastICA()
        ica.fit(data_scaled)

        # Print the independent components
        print ica.components_
```

```
[[ 0.00193392  0.07260048 -0.0550773  -0.00177143  0.01567576 -0.01707414]
 [-0.01093125 -0.00103289  0.00735043  0.05404891 -0.00265168 -0.01676763]
 [-0.00380912  0.01686284  0.11490758 -0.00708014 -0.13437409 -0.01615161]
 [ 0.05022515 -0.00631934 -0.00583923 -0.00328896  0.00972125 -0.00295345]
 [-0.0048809  -0.00161848 -0.00571174 -0.00253184  0.00243239  0.05096603]
 [ 0.00267466 -0.0139774   0.06025118  0.00204262 -0.00298658 -0.00399475]]
```

4) For each vector in the ICA decomposition, write a sentence or two explaining what sort of object or property it corresponds to. What could these components be used for?

Answer: Each vector corresponds to a linear combination of features. These components, or transformed features, identify fundamental features of your data that can be used for classification.

1.2 Clustering

In this section you will choose either K Means clustering or Gaussian Mixed Models clustering, which implements expectation-maximization. Then you will sample elements from the clusters to understand their significance.

1.2.1 Choose a Cluster Type

5) What are the advantages of using K Means clustering or Gaussian Mixture Models?

Answer:

- k-means clustering advantages
  - cheap relative to other unsupervised learning algorithms
  - scales well
- Gaussian mixture models advantages
  - Fastest mixture model algorithm
  - No bias of the means towards zero

We will use k-means since it is fast.

6) Below is some starter code to help you visualize some cluster data. The visualization is based on this demo from the sklearn documentation.

```
In [9]: # Import clustering modules
        from sklearn.cluster import KMeans
        from sklearn.mixture import GMM

In [10]: # First we reduce the data to two dimensions using PCA to capture variation
         reduced_data = PCA(n_components=2).fit_transform(data)
         print reduced_data[:10]  # print upto 10 elements
```

```
[[  -650.02212207   1585.51909007]
 [  4426.80497937   4042.45150884]
 [  4841.9987068    2578.762176  ]
 [  -990.34643689  -6279.80599663]
 [-10657.99873116  -2159.72581518]
 [  2765.96159271   -959.87072713]
 [   715.55089221  -2013.00226567]
 [  4474.58366697   1429.49697204]
 [  6712.09539718  -2205.90915598]
 [  4823.63435407  13480.55920489]]
```

In [14]:
```
# Implement your clustering algorithm here, and fit it to the reduced data for visualization
# The visualizer below assumes your clustering object is named 'clusters'

kmeans = KMeans()
kmeans.fit(reduced_data)

clusters = kmeans
print clusters
```

```
KMeans(copy_x=True, init='k-means++', max_iter=300, n_clusters=8, n_init=10,
    n_jobs=1, precompute_distances='auto', random_state=None, tol=0.0001,
    verbose=0)
```

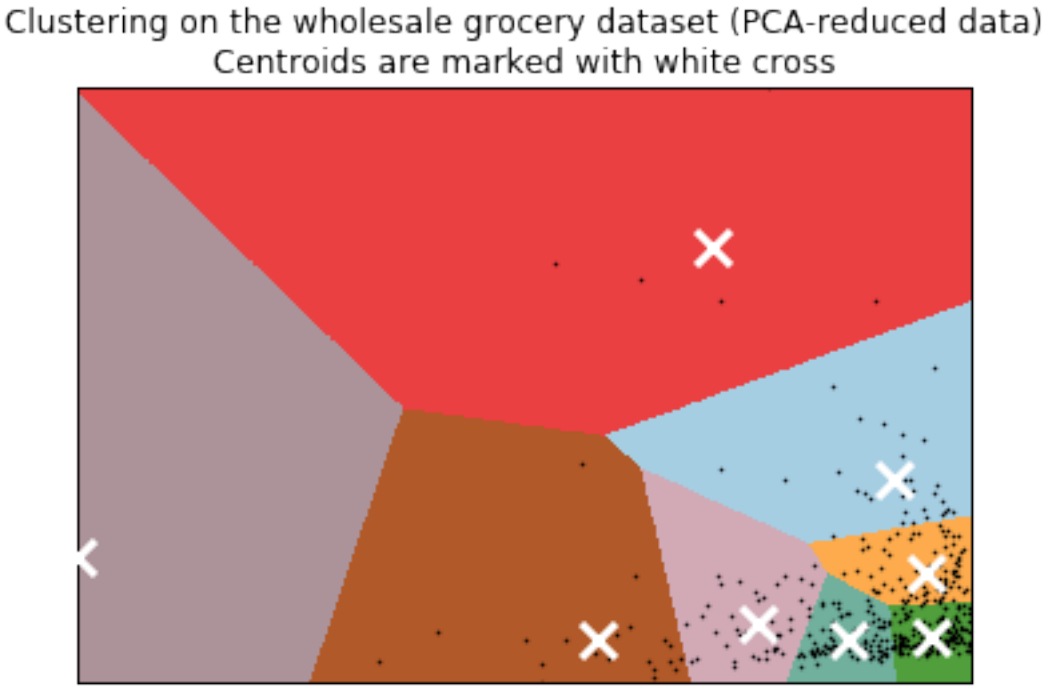
In [15]:
```
# Plot the decision boundary by building a mesh grid to populate a graph.
x_min, x_max = reduced_data[:, 0].min() - 1, reduced_data[:, 0].max() + 1
y_min, y_max = reduced_data[:, 1].min() - 1, reduced_data[:, 1].max() + 1
hx = (x_max-x_min)/1000.
hy = (y_max-y_min)/1000.
xx, yy = np.meshgrid(np.arange(x_min, x_max, hx), np.arange(y_min, y_max, hy))

# Obtain labels for each point in mesh. Use last trained model.
Z = clusters.predict(np.c_[xx.ravel(), yy.ravel()])
```

In [18]:
```
# Find the centroids for KMeans or the cluster means for GMM

centroids = kmeans.cluster_centers_
print centroids
```

```
[[   2787.06628956   24275.2601508 ]
 [  -2900.14423445   -6032.43569237]
 [   7855.70387893   -5357.47365718]
 [ -20964.64986166   68819.21772923]
 [   7012.62228466    6928.39601904]
 [ -15028.68476809   -2849.61643624]
 [-103863.42532004    9910.34962857]
 [ -35750.04892135   -6153.93703644]]
```

In [19]:
```
# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(1)
plt.clf()
plt.imshow(Z, interpolation='nearest',
           extent=(xx.min(), xx.max(), yy.min(), yy.max()),
           cmap=plt.cm.Paired,
           aspect='auto', origin='lower')

plt.plot(reduced_data[:, 0], reduced_data[:, 1], 'k.', markersize=2)
plt.scatter(centroids[:, 0], centroids[:, 1],
            marker='x', s=169, linewidths=3,
            color='w', zorder=10)
plt.title('Clustering on the wholesale grocery dataset (PCA-reduced data)\n'
          'Centroids are marked with white cross')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
```

```
plt.yticks(())
plt.show()
```



7) What are the central objects in each cluster? Describe them as customers.

Answer: The central objects in each cluster are the centroids. You can think of them as an average customer within that cluster.

1.2.2 Conclusions

** 8)** Which of these techniques did you feel gave you the most insight into the data?

Answer: Plotting the output of the k-means algorithm gave me the most insight into the data.

9) How would you use that technique to help the company design new experiments?

Answer: When designing new experiments, I would use only a small sample of customers from each cluster in my experiment.

10) How would you use that data to help you predict future customer needs?

Answer: I would use the experiment's result for the sample in each cluster to generalize and help predict future customer needs for all the customers in that cluster.