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 [54]:

```
# Import libraries: NumPy, pandas, matplotlib
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 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
```

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: By looking at the first five rows of the provided dataset, it appears the the feature with the most influence will be the same as the feature with the highest average spend and highest standard deviation, which in this case is the "Fresh" category. It appears, therefore, that the first PCA dimension will be most heavily influenced by the "Fresh" features. Because of the similarity between data variance and standard deviation, I will assume that the figures with the highest standard deviation will be the most influential on PCA analysis.

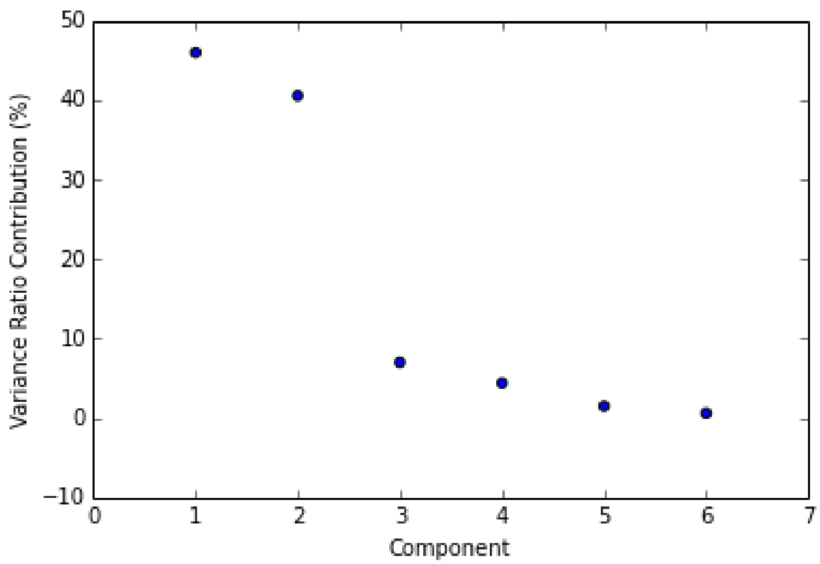
PCA

In [55]:

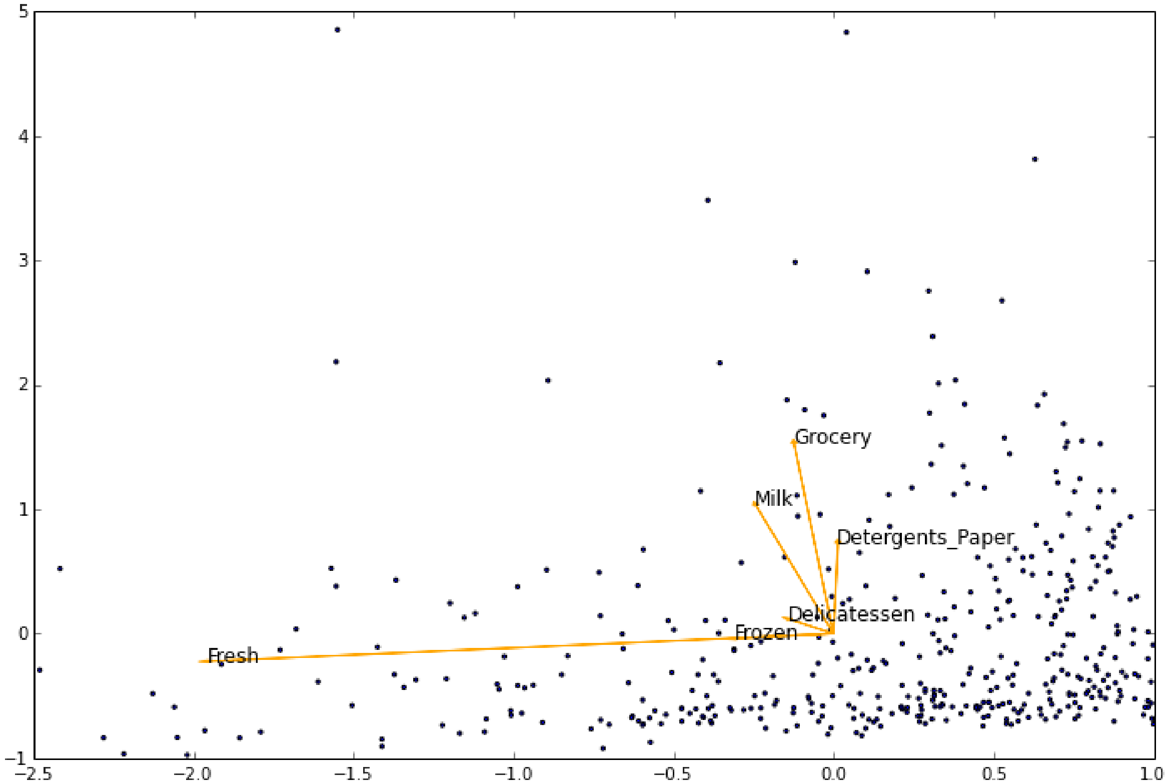
```
# TODO: Apply PCA with the same number of dimensions as variables i
n the dataset


from sklearn.decomposition import PCA
pca = PCA(n_components = 6).fit(data)
comp_var = 0
i = 1
var_contribution = {}
print "Variance Ratio Contribution and Percent Increase By Componen
t "
for variance in  pca.explained_variance_ratio_:
    prev_var = comp_var
    comp_var = comp_var+variance
    increase_per = round((comp_var-prev_var)/prev_var * 100,3)
    display_var = round(float(comp_var*100), 3)
    var_contribution[i] = variance*100
    print "Component " + str(i) + " - " + "Variance Ratio Total: "
+ str(display_var) + "% | Percentage increase: " + str(increase_pe
r) + "%"
    i += 1

plt.scatter(var_contribution.keys(), var_contribution.values())
plt.xlabel('Component')
plt.ylabel('Variance Ratio Contribution (%)')
plt.show()

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


print "\n\n ################### Biplot below ################ \n
\n"

# Biplot function, see reference in answer to question 3


# Code snippet taken from Udacity Forums and modified, reference li
nk below:
# https://discussions.udacity.com/t/having-trouble-with-pca-and-ica
-specifically-with-explaining-what-the-dimensions-mean/41890/11
def pca_biplot(new_data):
    # Fit on 2 components
    pca2 = PCA(n_components=2, whiten=True).fit(new_data)
    print pca2.components_
    # Plot transformed/projected data
    format_data = pd.DataFrame(
        pca2.transform(new_data),
        columns=['PC1', 'PC2']
    )

    plt.figure(figsize=(12, 8))
```

```
    plt.scatter(format_data["PC1"], format_data["PC2"], marker='.')

    plt.xlim(-2.5,1.0)
    plt.ylim(-1.0,5)
    # Plot arrows and labels
    for i, (pc1, pc2) in enumerate(zip(pca2.components_[0]*2, pca2.
components_[1]*2)):
        plt.arrow(0, 0, pc1, pc2, width=0.001, fc='orange', ec='ora
nge')
        plt.annotate(new_data.columns[i], (pc1, pc2), size=12)
    plt.show()


pca_biplot(data)
```

```
Variance Ratio Contribution and Percent Increase By Component
Component 1 - Variance Ratio Total: 45.961% | Percentage increas
e: inf%
Component 2 - Variance Ratio Total: 86.479% | Percentage increas
e: 88.155%
Component 3 - Variance Ratio Total: 93.482% | Percentage increas
e: 8.098%
Component 4 - Variance Ratio Total: 97.884% | Percentage increas
e: 4.709%
Component 5 - Variance Ratio Total: 99.386% | Percentage increas
e: 1.535%
Component 6 - Variance Ratio Total: 100.0% | Percentage increase:
0.618%
```




```
[[-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  0.0
0613848]


 ################### Biplot below ################


[[-0.97653685 -0.12118407 -0.06154039 -0.15236462  0.00705417 -0.
06810471]
 [-0.11061386  0.51580216  0.76460638 -0.01872345  0.36535076  0.
05707921]]
```



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?

Answer: The variance contribution appears to be something of a step function with the majority of the contribution, 87.47% in total, coming from the first two dimensions. The next four dimensions have very small contributions that appear to be decreasing in a negative exponential fashion.

Depending upon tests for efficiency and increased accuracy, I would choose either 2 or 3 dimensions. After the third dimension each subsequent dimension contributes 3.8% or less to the total variance ratio.

Forced to pick one value without further testing, I would pick 2 dimensions, covering 86.47% of the total variance ratio.

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

Answer: To answer this question I used one external resource

1) Udacity Forums - https://discussions.udacity.com/t/having-trouble-with-pca-and-ica-specifically-with-explaining-what-the-dimensions-mean/41890/11 (https://discussions.udacity.com/t/having-trouble-with-pca-and-ica-specifically-with-explaining-what-the-dimensions-mean/41890/11) This put me in the right direction to visualize the dimensions using the bi-plot above.

Using the statistics within the README file, the biplot confirms my initial hypothesis that the features with the highest standard deviation would be the largest contributors to PCA dimensions. In this case the first dimension appears to be the Fresh category while the second component is most heavily influenced by the Grocery category.

ICA

In [56]:

```
# TODO: Fit an ICA model to the data
# Note: Adjust the data to have center at the origin first!
from sklearn.decomposition import FastICA
std_data = data
std_data /= std_data.std(axis=0)
```

```
In [58]: ica = FastICA(n_components=6, whiten=True).fit(std_data)
         ica_trans = ica.transform(std_data)
         # Print the independent components

         print "\n ######## Components ########## \n"
         print np.round_(ica.components_*100, decimals=6)
         print "\n ######## Mixing Matrix ########## \n"
         print np.round_(ica.mixing_, decimals=4)
```

```
 ######## Components ##########

[[  0.299846   1.015974   8.559058  -0.189822  -6.919686  -4.7548
83]
 [ -0.384875   2.615712   3.170223  -0.638645  -8.901375  -0.9494
]
 [ -0.274918  -4.964186  10.656336  -0.244749  -6.871597   2.6321
74]
 [ -0.104256   5.01633    1.365263  -0.290099  -2.914386   0.9378
29]
 [ -5.047973   0.588754   0.917877   0.316188  -1.293204   0.3969
3 ]
 [  1.080571   0.214982  -0.875057  -5.416625   0.321099   1.7778
47]]

 ######## Mixing Matrix ##########

[[ -0.9796   1.294    1.5795   3.556  -20.4633  -1.5746]
 [  3.269   -6.4535  -3.2437  19.3887   0.4151  -0.2213]
 [  8.4629 -14.5753   5.4479  11.1053   1.2276   0.6526]
 [ -5.5004   0.9866   2.4108   4.3501  -4.6904 -19.001 ]
 [  5.8461 -18.3567  -0.2682   7.8214   1.9567   1.6669]
 [-13.4487  -0.8588   9.5071  12.8012  -1.6523  -0.6391]]
```

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:

I'm not completely sure how to interpret each vector in the matrix. It appears that each vector has a dominant component, for instance vector 1 has a dominant component in position 3, vector 2 in position 5, etc.

However, I'm unsure on how to interpret these results, as well as if each figure is solely a magnitude value or if these are not on an absolute scale. More resources on ICA would be very helpful, I've watched all videos twice, and read a dozen articles but expanding the general signal separation problem using ICA to an n-dimensional data analysis is escaping me.

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.

Choose a Cluster Type

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

Answer:

K-Means clustering is a general purpose algorithm that scales easily to large sets of data.

Gaussian Mixture Model, which is a soft clustering method, works with any distribution and is extremely fast.

6) Below is some starter code to help you visualize some cluster data. The visualization is based on [this demo (http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html)](http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html) from the sklearn documentation.

In [61]:

```
# Import clustering modules
from sklearn.cluster import KMeans
from sklearn.mixture import GMM
```

In [62]:

```
# TODO: First we reduce the data to two dimensions using PCA to cap
ture variation
reduced_data = PCA(n_components=2, whiten=True).fit_transform(data)
print reduced_data[:10]  # print upto 10 elements
```

```
[[-0.05066239  0.13161505]
 [ 0.34502287  0.33556674]
 [ 0.37738285  0.21406486]
 [-0.07718708 -0.5212911 ]
 [-0.83067886 -0.17928035]
 [ 0.2155776  -0.07967954]
 [ 0.05576966 -0.16710073]
 [ 0.34874672  0.11866355]
 [ 0.52313722 -0.18311407]
 [ 0.37595155  1.11903068]]
```

In [64]:

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

clusters = KMeans(n_clusters=3).fit(reduced_data)
```

In [65]:

```
# 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()])
```

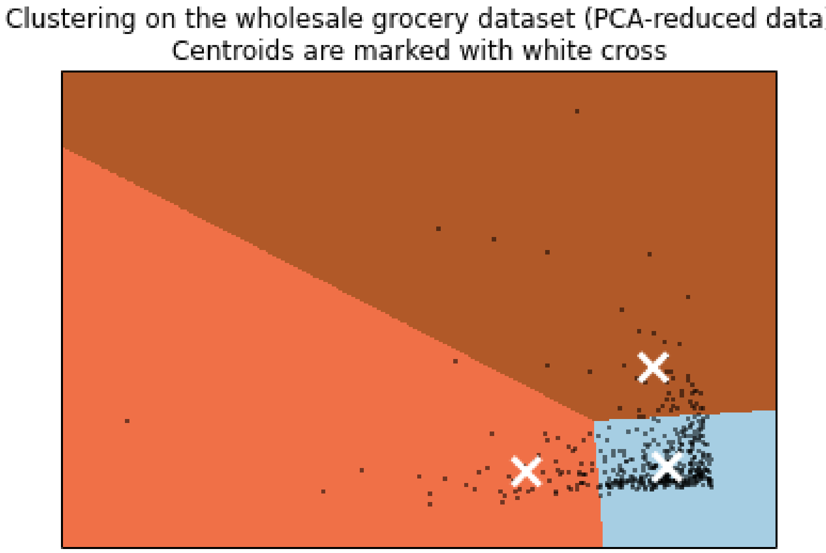
In [66]:

```
# TODO: Find the centroids for KMeans or the cluster means for GMM

centroids = clusters.cluster_centers_
print centroids
```

```
[[ 0.32398252 -0.25421161]
 [-1.86890029 -0.36902956]
 [ 0.10439573  2.12063212]]
```

In [28]:

```
# 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: Starting with the blue cluster, proceeding to the dark red, and then to the light red.

The first cluster appears to be the majority of customers who purchase Fresh and Grocery in nearly equal proportion with a slight bias toward Fresh.

The second cluster appears to be customers who are much more heavily biased towards purchasing groceries with very little correlation in their behavior and the Fresh category.

Conversely, the third cluser appears to be the opposite of the second, with customers highly biased toward the Fresh category with even lower correlation to the Grocery category.

Conclusions

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

Answer: The PCA analysis when combined with the bi-plot helped me visualize what was going on. I am still not completely solidified on any of the topics covered in this project and would very much appreciate additional resources.

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

Answer: I'm finding this question a little ambiguous, are we creating more ML experiments, or business logic experiments such as testing marketing budget allocation to a product line? Please clarify.

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

Answer: The data appears to describe a consumption trend heavily influenced by the fresh category when the Grocery category for the customer is below a certain threshold. You could infer from this that stores with Grocery category orders below a certain threshold will be more likely to have a large order in the Fresh category.