

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 [1]:

```
# 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: I guess Fresh and Milk will come out the top two as they are both fast moving goods that consumers buy regularly.

###PCA

In [27]:

```
# TODO: Apply PCA with the same number of dimensions as variables in the dataset
from sklearn.decomposition import PCA
pca = PCA(n_components=6).fit(data)

# Print the components and the amount of variance in the data contained in each component
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?

Answer: The first two features take up close to 0.86 of the variances (in ratio) and the remaining features are insignificant with all below 0.07. If i were to use PCA on this dataset, I will choose 2 dimensions for my analysis.

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

Answer:

The dimensions represents the eigenvalues that corespond to each of the largest eiganvector for the corresponding features.I can use the information to select the most significant features by choosing those corresponding to the largest eigenvalues shown.

###ICA

In [3]:

```
# TODO: Fit an ICA model to the data
# Note: Adjust the data to have center at the origin first!
from sklearn.decomposition import FastICA
from sklearn.preprocessing import MinMaxScaler
data_scaled = MinMaxScaler().fit_transform(data)
ica = FastICA().fit(data_scaled)

# Print the independent components
print ica.components_

[[ 0.01783618  0.71634119 -0.5822624  -0.01922108  0.17909906 -0.
27677487]
 [ 0.04355432  0.01852258  0.05699      0.03213772 -0.02277868 -0.
86918239]
 [ 0.44627664 -0.06644818 -0.03640058 -0.03998374  0.08101431 -0.
0528785 ]
 [-0.03301235  0.238879      0.57527654 -0.08816343 -0.92943472 -0.
19759561]
 [ 0.01115648  0.00172861 -1.10842589  0.03373713  0.67059151  0.
21252628]
 [ 0.09748927  0.01020347 -0.07930122 -0.67805467  0.0265216   0.
28605527]]

/Users/junhua_liu/Playground/udacity/MLND/venv/p1/lib/python2.7/s
ite-packages/scipy/linalg/basic.py:884: RuntimeWarning: internal
gelsd driver lwork query error, required iwork dimension not retu
rned. This is likely the result of LAPACK bug 0038, fixed in LAPA
CK 3.2.2 (released July 21, 2010). Falling back to 'gelss' drive
r.
  warnings.warn(mesg, RuntimeWarning)
```

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:

Label: 1.Fresh 2.Milk 3.Grocery 4.Frozen 5.Detergents_Paper 6.Delicatessen

Objects corresponding to each vector as follows:

2,6,3,1,5,4

##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 is much more scalable than GMM as it can take very large `n_samples`. GMM has a flat geometry which performs better for density estimation as compared to K Means which has a non-flat geometry.

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 [4]:

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

In [47]:

```
# TODO: First we reduce the data to two dimensions using PCA to capture variat
pca = PCA(n_components=2).fit(data)
```

```
reduced_data = pca.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 [97]:

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

```
# clusters = KMeans(n_clusters=3).fit(reduced_data)
clusters = GMM(n_components=4).fit(reduced_data)
print clusters
```

```
GMM(covariance_type='diag', init_params='wmc', min_covar=0.001,
    n_components=4, n_init=1, n_iter=100, params='wmc', random_stat
e=None,
    thresh=None, tol=0.001, verbose=0)
```

In [98]:

```
# 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 [99]:

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

```
centroids = clusters.means_
# centroids = clusters.cluster_centers_
print centroids
```

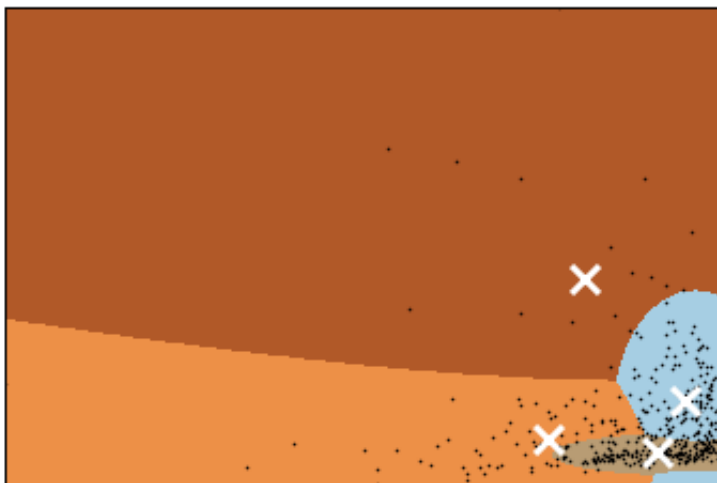
```
[[ 7174.54719282   5469.02876453]
 [ 2339.15204219 -6708.93065712]
 [-15372.37194307 -3334.43379857]
 [-9486.9742574   34645.20428228]]
```

In [100]:

```
# 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()
```

Clustering on the wholesale grocery dataset (PCA-reduced data)
Centroids are marked with white cross



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

Answer: centroids represent different types of customers with various purchasing patterns.

###Conclusions

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

Answer: GMM

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

Answer: one example of new experiments is to analyse customer's sensibility to price change. By changing the price of different categories of goods, we can observe the change in the purchasing pattern.

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

Answer: I can further analyse for the different groups of customers, which groups are more profitable and what categories of goods they buy most frequently. Using this information I can add variety or increase the price for those categories to hopefully increase sales and revenue.