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 [27]:
```
# 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: Just looking at the variation by considering the min and max of the table, we could definitely observe a dimension with capturing high variation of "Fresh","Milk", and "Grocery". The variance along the principal components play a large role in selecting dimensions. The PCA dimensions tries to mutually orthogonal, maximize variance and has an ordered set of features.

Ica helps identify independent components by maximizing independence of the estimated components. Therefore ICA vectors are vectors in an matrix called 'unmixing' matrix which assumes independence among the dimensions and gives the best possible matrix to reconstruct the information. ICA dimensions are therefore mutually independent, maintains maximal mutual information and has a 'bag' of features. Doesn't really care about the order of the features.

ICA, the goal is to separate superimposed signals which basically comes from highest and family run low volume customers in this context. So, the vectors we have are matrices we will use to transform the dataset we have to our hidden or source variables, the two types of customers we are interested to identify.

###PCA

In [37]:
```
# TODO: Apply PCA with the same number of dimensions as variables in th
from sklearn.decomposition import PCA
pca = PCA(n_components=6)
pca.fit(data)

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

```
[[-0.97653685 -0.12118407 -0.06154039 -0.15236462  0.00705417 -0.068
10471]
 [-0.11061386  0.51580216  0.76460638 -0.01872345  0.36535076  0.057
07921]
 [-0.17855726  0.50988675 -0.27578088  0.71420037 -0.20440987  0.283
21747]
 [-0.04187648 -0.64564047  0.37546049  0.64629232  0.14938013 -0.020
39579]
 [ 0.015986    0.20323566 -0.1602915   0.22018612  0.20793016 -0.917
07659]
 [-0.01576316  0.03349187  0.41093894 -0.01328898 -0.87128428 -0.265
41687]]
[ 0.45961362  0.40517227  0.07003008  0.04402344  0.01502212  0.0061
3848]
```

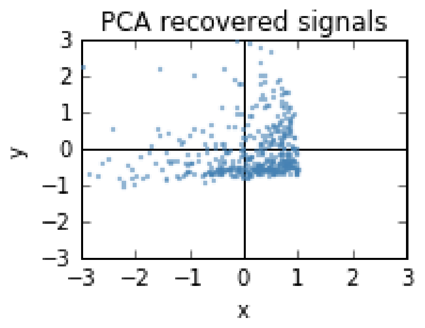
2) How quickly does the variance drop off by dimension? If you were to use PCA on this dataset, how many dimensions would you choose for your analysis? Why?

Answer: After the first two dimensions, variance drop off considerably. However, the first four dimensions contribute to 97.8% of total variance. I would use four and could be confident that we are minimizing the loss of information on the original variables.

In [4]:
```
def plot_samples(S, axis_list=None):
    plt.scatter(S[:, 0], S[:, 1], s=2, marker='o', zorder=10,
                color='steelblue', alpha=0.5)
    if axis_list is not None:
        colors = ['orange', 'red']
        for color, axis in zip(colors, axis_list):
            axis /= axis.std()
            x_axis, y_axis = axis
            # Trick to get legend to work
            plt.plot(0.1 * x_axis, 0.1 * y_axis, linewidth=2, color=col
            plt.quiver(0, 0, x_axis, y_axis, zorder=11, width=0.01, sca
                       color=color)

    plt.hlines(0, -3, 3)
    plt.vlines(0, -3, 3)
    plt.xlim(-3, 3)
    plt.ylim(-3, 3)
    plt.xlabel('x')
    plt.ylabel('y')
```

In [5]:
```
#Plotting PCA variance

S_pca_ = pca.fit(data).transform(data)
plt.subplot(2, 2, 3)
plot_samples(S_pca_ / np.std(S_pca_, axis=0))
plt.title('PCA recovered signals')
```

Out[5]:
```
<matplotlib.text.Text at 0x110bbca90>
```



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

Answer: The dimensions represent how best they could maximize the variation along the orthogonal axis and therefore minimize the loss of information. In case of 'curse of dimensionality', we could employ PCA to do a 'dimensionality reduction' while preserving information as best as possible. As a result, the computation becomes faster and efficient.

For the First basis vector, we can see variations of "Fresh" is captured more than any others. Milk and Frozen are next. Seems like it could be low volume pop-and-mom shops, grocery stores,etc.

For the second basis vector, we can see maximum variation is captured on Milk, Grocery and Detergents_Paper.

For the sixth basis vector, we can see "Detergents_paper" and "Delicatessen" variations are captured much better than the others. It reminds me of shops like Costco and Sam's club and therefore could represent high volume customers.

In PCA,we try to take advantage of reducing the dimensionality. So, We could then take these dimensions, fit our data to these dimensions (while minimizing informaton loss by definition) and then apply clustering techniques on top of fitted data.

Since, the goal of the project is to see what different types of customers we have, the explained variance ratio on each of these dimensions give insight into the spending pattern behavior of customers on each category. The number of clusters could capture the number of different spending behaviors exhibited by the customers.

###ICA

In [38]:
```
# TODO: Fit an ICA model to the data
# Note: Adjust the data to have center at the origin first!
from sklearn.decomposition import FastICA
ica = FastICA(n_components=6)
data /= data.std(axis=0)
ica.fit(data)

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

```
[[-0.00194681 -0.07265135  0.05525287  0.00176264 -0.01586952  0.017
07672]
 [ 0.00377522 -0.01710695 -0.11436054  0.00710871  0.13445839  0.016
15971]
 [ 0.00488818  0.00162034  0.00570189  0.00253453 -0.00242956 -0.051
02219]
 [-0.05028636  0.00635009  0.00600001  0.00328579 -0.00990018  0.002
93155]
 [-0.00265371  0.01387362 -0.06148114 -0.00196939  0.00431968  0.004
15736]
 [ 0.01094161  0.00103816 -0.0073547  -0.05411133  0.00264114  0.016
78544]]
```

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: ICA gives the unmixing matrix to identify the latent or source variables, taking Blind Source Separation problem for example. In our case, the sources are the different types of customers separated or clustered based on the spending volume of the customers. Therefore the ICA matrix could help separate those customers into appropriate customers.

The direction positive or negative could be considered as expression levels. ICA tries to produce a component that maximizes the independence of the sources. So, a positive number could be considered as a positive expression level and negative number is considered as a negative expression level.

On the first vector, we see signals of a customers who stock more Grocery, Frozen and Delicatessan and less on others.

On the second, we see signals of a customers who stock more on Fresh, Frozen, Detergent Paper, Delicatessan and less on Milk and Grocery. Looks like a high volume/supermarket scenario.

On the third, We see signals of a customers who stock less on Detergent Paper and Delicatessan but equally more on others. Looks like a low volume/ordinary grocery scenario.

On the fourth, We see signals of a customers who stock less on Detergent Paper and Fresh but more on others, especially on Milk and Grocery. Looks like another low volume/ordinary grocery scenario.

##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 a. compuationally efficient b. runs usually fast - converges using local minimum c. cluster scatter is calculated as a measure of eucladian distance

```
    GMM is
     a. it is the fastest algorithm for learning mixture model
s
     b. It can also draw confidence ellipsoids for multivariat
e models,
     c. and compute the Bayesian Information Criterion to asse
ss the number of clusters in the data.            S
```

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) from the sklearn documentation.

In [7]:
```
# Import clustering modules
from sklearn.cluster import KMeans
from sklearn.mixture import GMM
```

In [8]:
```
# TODO: First we reduce the data to two dimensions using PCA to capture
reduced_data = PCA(n_components=2).fit_transform(data)
print reduced_data[:10]  # print upto 10 elements
```

```
[[-0.19307077  0.30475306]
 [-0.43392596  0.32803921]
 [-0.81022096 -0.81416893]
 [ 0.7777625  -0.65201155]
 [-0.16609819 -1.26998809]
 [ 0.15599237  0.29480541]
 [ 0.33490718  0.52440632]
 [-0.14042659  0.23073005]
 [ 0.51673134  0.65861312]
 [-1.59029884  0.74016879]]
```

In [20]:
```
# TODO: Implement your clustering algorithm here, and fit it to the red
# The visualizer below assumes your clustering object is named 'cluster

clusters = KMeans(init='k-means++', n_clusters=2, n_init=10)#?
clusters.fit(reduced_data)
print clusters
```

```
KMeans(copy_x=True, init='k-means++', max_iter=300, n_clusters=2, n_
init=10,
    n_jobs=1, precompute_distances='auto', random_state=None, tol=0.
0001,
    verbose=0)
```

In [21]:
```
lowest_bic = np.infty
bic = []
n_components_range = range(1, 7)
cv_types = ['spherical', 'tied', 'diag', 'full']
for cv_type in cv_types:
    for n_components in n_components_range:
        # Fit a mixture of Gaussians with EM
        gmm = GMM(n_components=n_components, covariance_type=cv_type)
        gmm.fit(reduced_data)
        bic.append(gmm.bic(reduced_data))
        if bic[-1] < lowest_bic:
            lowest_bic = bic[-1]
            best_gmm = gmm
clusters=best_gmm
print clusters
```

```
GMM(covariance_type='full', init_params='wmc', min_covar=0.001,
  n_components=5, n_init=1, n_iter=100, params='wmc', random_state=N
one,
  thresh=None, tol=0.001, verbose=0)
```

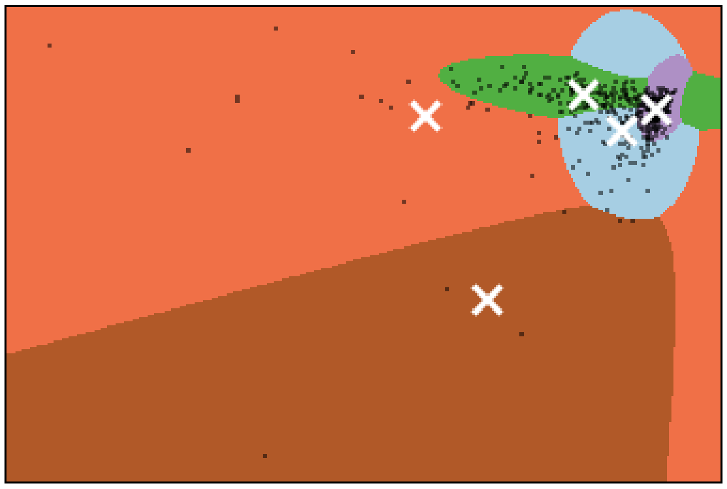
In [22]:
```
# Plot the decision boundary by building a mesh grid to populate a grap
x_min, x_max = reduced_data[:, 0].min() - 1, reduced_data[:, 0].max() +
y_min, y_max = reduced_data[:, 1].min() - 1, reduced_data[:, 1].max() +
hx = (x_max-x_min)/1000.
hy = (y_max-y_min)/1000.
xx, yy = np.meshgrid(np.arange(x_min, x_max, hx), np.arange(y_min, y_ma

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

In [23]:
```
# TODO: Find the centroids for KMeans or the cluster means for GMM

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

```
[[ 0.9805629   0.07638532]
 [-2.89370972 -7.73599815]
 [-4.21790443 -0.09577624]
 [ 0.21209092 -0.77742979]
 [-0.62447778  0.73952784]]
```

In [13]:
```
# 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 dat
          '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: There are five clusters are they are visible clearly. In each cluster, we have a set of customers whose spending habits are closer to the centroids (which are the central objects) of the cluster.

###Conclusions

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

Answer: GMM gave the most insight into the data. The main advantage we have is the abilitiy to determine the number of components in an efficient way using BIC Criterion which I also implemented and figured out the best cluster parameters to use.

It is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians. Taken from http://scikit-learn.org/stable/modules/mixture.html#pros (http://scikit-learn.org/stable/modules/mixture.html#pros)

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

Answer: A/B testing allows individuals, teams, and companies to make careful changes to their user experiences while collecting data on the results.

Using the above technique, we can now have five different clusters or segments of customers. Instead of implementing a completely different delivery option on all customers, we will treat the segements independently and test the delivery changes using A/B testing method on each group.

Depends on the response for each control group, we will decide whether to rollout the delivery options to the entire segement of the customers. We will repeat the process for all five customer segements.

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

Answer: Having decided that we are going to treat the identified segments independently, we can build supervised techniques like randomForest, DecisionTree to build a model with the target being the number of items sold in each category (derived from total spend).

Once we have built the models,we can split them as train and test sets, fit a model for every category of food (mnilk, fresh,etc) over evey single segment then use the test to predict the demand.

Once have build models with sufficient accuracy, we can then group a set of new customers as a test set, identify what cluster they will belong in (unsupervised) and then apply the correct supervised trained model to make the prediction. The key lesson is that we will have different supervised model for each segmentation.