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 [296]:
```
# 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 tries to produce components that maximizes the separation between separate classes. From the notes, we see Kurtosis which is the spikiness or peakedness is used to separate the classes. The classes in our case could be the different customers sets we have - based on high volume or low volume spend. Therefore a simple frequency distribution of anual spend 'range' over categories could show any spike or curtosis we are looking for.

For low volume or family run shops, I would expect to a spike selling fresh and groceries (Fresh, Milk and Grocery) and specially prepared foods, ("Delicatessen').

For high volume customers like Sams club, I would expect them to sell more Frozen and Detergents and Paper products. I don't usually think about Milk and Fresh and Delicatessen from high volume customers.

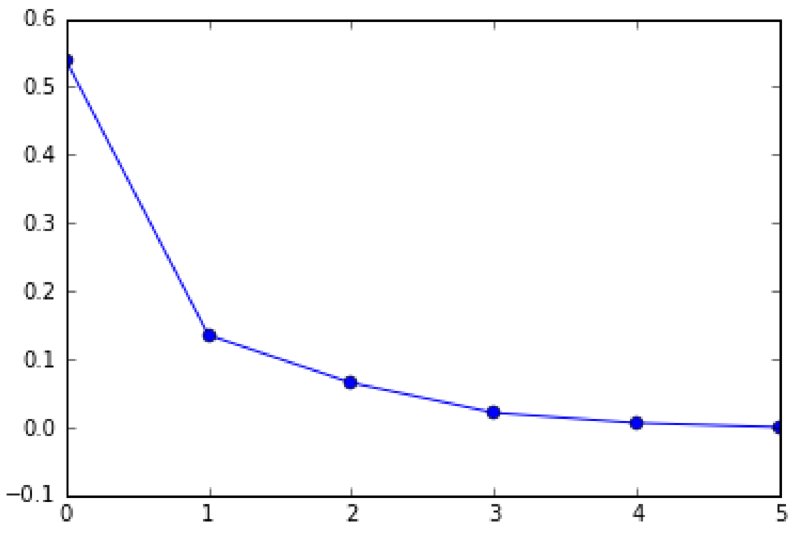
So, based on the spikes or curtosis I am expecting from above theories on those two different classes, I would expect two dimensions that maximizes the separation between the class of the customers who are the source of the distribution of the data we have.

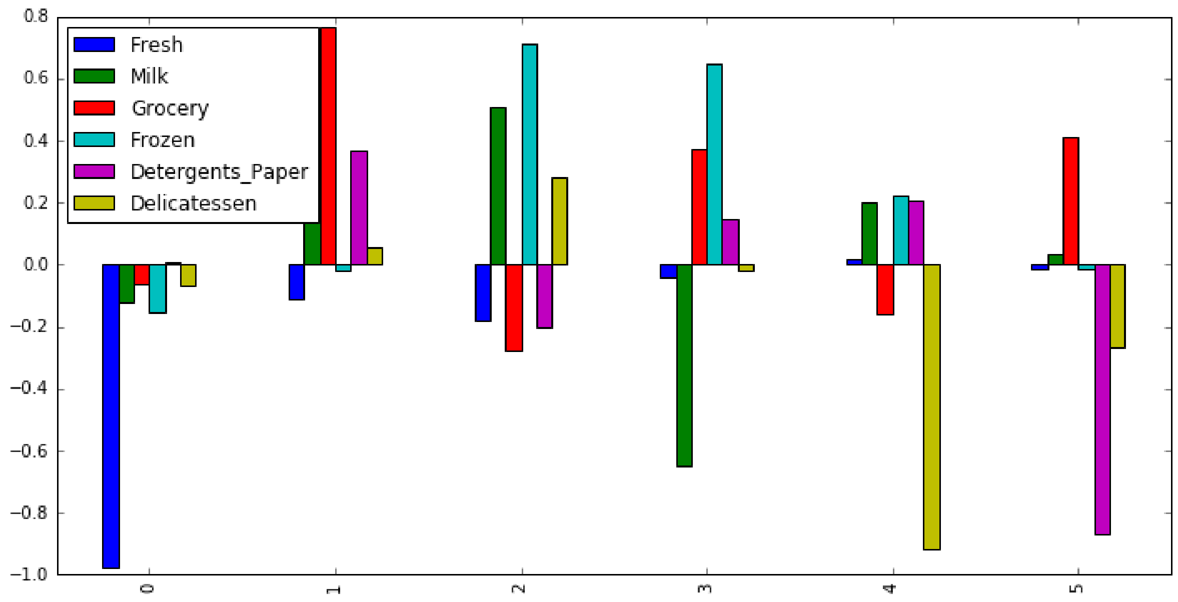
PCA

```
In [297]: # 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)
          pca.fit(data)

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

          x = np.arange(6)
          plt.plot(x, 1 - np.cumsum(pca.explained_variance_ratio_), '-o')
          pd.DataFrame(pca.components_,columns=data.columns).plot(kind='bar',
          figsize = (12,6))
```

```
[[-0.97653685 -0.12118407 -0.06154039 -0.15236462  0.00705417 -0.0
6810471]
 [-0.11061386  0.51580216  0.76460638 -0.01872345  0.36535076  0.0
5707921]
 [-0.17855726  0.50988675 -0.27578088  0.71420037 -0.20440987  0.2
8321747]
 [-0.04187648 -0.64564047  0.37546049  0.64629232  0.14938013 -0.0
2039579]
 [ 0.015986    0.20323566 -0.1602915   0.22018612  0.20793016 -0.9
1707659]
 [-0.01576316  0.03349187  0.41093894 -0.01328898 -0.87128428 -0.2
6541687]]
[ 0.45961362  0.40517227  0.07003008  0.04402344  0.01502212  0.00
613848]
```

```
Out[297]: <matplotlib.axes._subplots.AxesSubplot at 0x117949810>
```

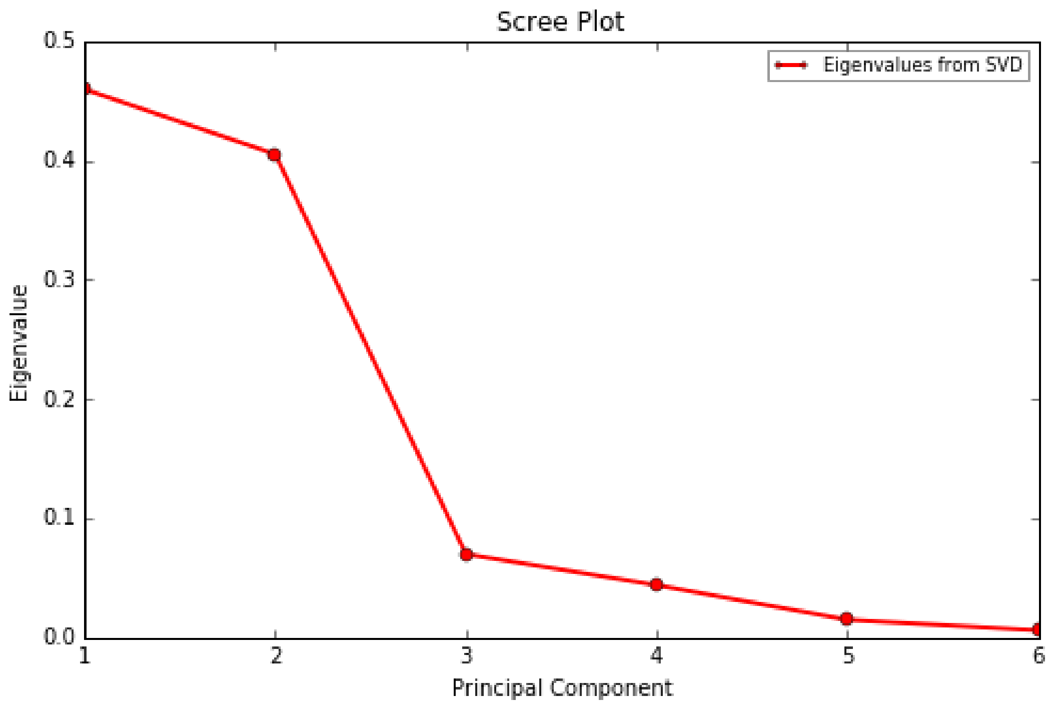


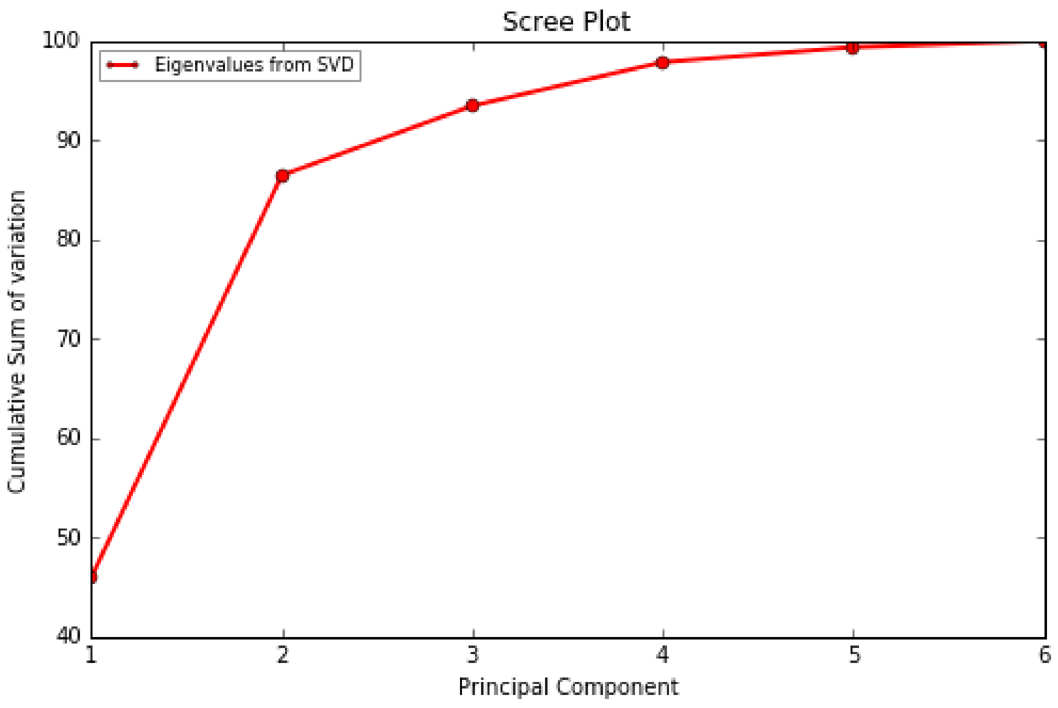


In [298]:

```
#!/usr/bin/env python
# coding: utf8
# GistID: 8785541

from __future__ import division

import numpy as np
import matplotlib
import matplotlib.pyplot as plt

# Make a random array and then make it positive-definite
A = np.random.randn(6, 9)
A = np.asmatrix(A) * np.asmatrix(A.T)
U, S, V = np.linalg.svd(A)
eigvals = S**2 / np.cumsum(S)[-1]
eigvals2 = S**2 / np.sum(S)
assert (eigvals == eigvals2).all()

eigvals = pca.explained_variance_ratio_
tot = sum(eigvals)
var_exp = [(i / tot)*100 for i in sorted(eigvals, reverse=True)]
cum_var_exp = np.cumsum(var_exp)

fig = plt.figure(figsize=(8,5))
sing_vals = np.arange(len(eigvals)) + 1
plt.plot(sing_vals, eigvals, 'ro-', linewidth=2)
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Eigenvalue')
#I don't like the default legend so I typically make mine like belo
w, e.g.
#with smaller fonts and a bit transparent so I do not cover up dat
a, and make
#it moveable by the viewer in case upper-right is a bad place for i
t
leg = plt.legend(['Eigenvalues from SVD'], loc='best', borderpad=0.
3,
                 shadow=False, prop=matplotlib.font_manager.FontPro
perties(size='small'),
                 markerscale=0.4)
leg.get_frame().set_alpha(0.4)
leg.draggable(state=True)
plt.show()

#---
fig = plt.figure(figsize=(8,5))
sing_vals = np.arange(len(cum_var_exp)) + 1
plt.plot(sing_vals, cum_var_exp, 'ro-', linewidth=2)
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Cumulative Sum of variation')
#I don't like the default legend so I typically make mine like belo
w, e.g.
#with smaller fonts and a bit transparent so I do not cover up dat
```

```
a, and make
#it moveable by the viewer in case upper-right is a bad place for i
t
leg = plt.legend(['Eigenvalues from SVD'], loc='best', borderpad=0.
3,
                 shadow=False, prop=matplotlib.font_manager.FontPro
perties(size='small'),
                 markerscale=0.4)
leg.get_frame().set_alpha(0.4)
leg.draggable(state=True)
plt.show()

def main():
    """Run main."""

    return 0

if __name__ == '__main__':
    main()
```





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: I plotted both the eigen values and their explained variation of each of the PCA dimension using "scree plot" concept as explained in http://support.minitab.com/en-us/minitab/17/topic-library/modeling-statistics/multivariate/principal-components-and-factor-analysis/what-is-a-scree-plot/ (http://support.minitab.com/en-us/minitab/17/topic-library/modeling-statistics/multivariate/principal-components-and-factor-analysis/what-is-a-scree-plot/) and https://plot.ly/ipython-notebooks/principal-component-analysis/#PCA-Vs.-LDA (https://plot.ly/ipython-notebooks/principal-component-analysis/#PCA-Vs.-LDA)

These are my observations.

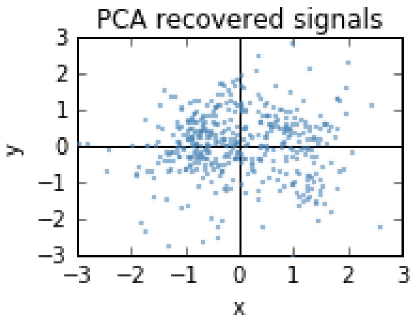
There is a steep drop in variation after two dimensions. The first two contribue to nearly 85% of the variation. However, based on the scree plot criteria, we will retain all the dimensions before the point that starts the flat line trend i.e. no more increase in the variation.

Looking at the second graph that shows the cumulative variation by dimension, we see the flat trend starts to happens at the fifth vector. Based on the guideline provided by scree, I will use four dimenision for analyis.

I also tried to see if I can use the Kaiser method as explained in http://support.minitab.com/en-us/minitab/17/topic-library/modeling-statistics/multivariate/principal-components-and-factor-analysis/number-of-principal-components/ (http://support.minitab.com/en-us/minitab/17/topic-library/modeling-statistics/multivariate/principal-components-and-factor-analysis/number-of-principal-components/) but normalization of eigenvalues in scikit eigenvector cannot use the rule to throw out anything having value greater than 1.0.

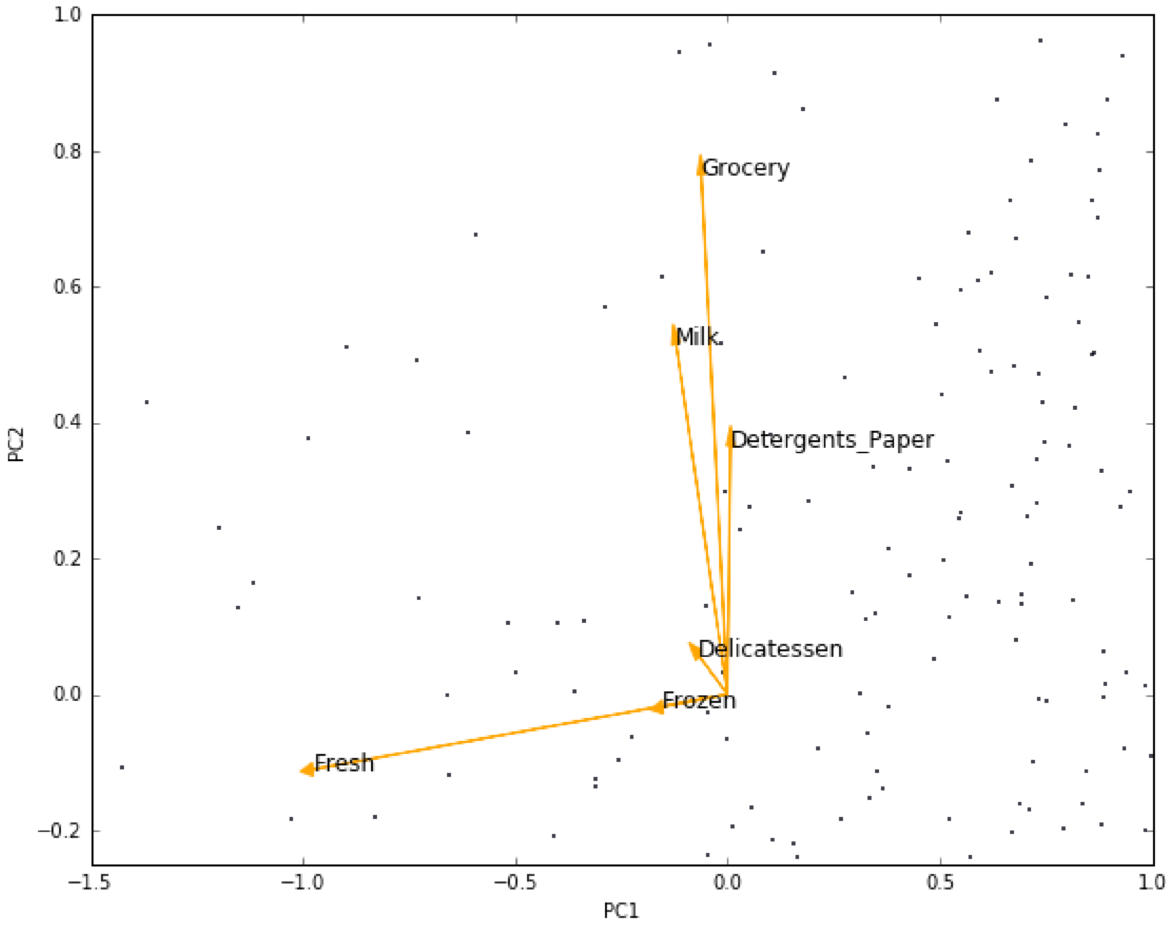
In [299]:
```
def plot_samples(S, axis_list=None):
    plt.scatter(S[:, 0], S[:, 1], s=2, marker='o', zorder=10,
                color='steelblue', alpha=0.5)
    if axis_list is not None:
        colors = ['orange', 'red']
        for color, axis in zip(colors, axis_list):
            axis /= axis.std()
            x_axis, y_axis = axis
            # Trick to get legend to work
            plt.plot(0.1 * x_axis, 0.1 * y_axis, linewidth=2, color
=color)
            plt.quiver(0, 0, x_axis, y_axis, zorder=11, width=0.01,
scale=6,
                       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 [300]:
```
#Plotting PCA variance
#data /= data.std(axis=0)
S_pca_ = pca.fit(np.log(data)).transform(np.log(data))
plt.subplot(2, 2, 4)
plot_samples(S_pca_ / np.std(S_pca_, axis=0))
plt.title('PCA recovered signals')
```
Out[300]:
```
<matplotlib.text.Text at 0x118eee790>
```



In [301]:
```
import pandas as pd
from sklearn.decomposition import PCA

def biplot(df):
    # Fit on 2 components
    pca = PCA(n_components=2, whiten=True).fit(df)

    # Plot transformed/projected data
    ax = pd.DataFrame(
        pca.transform(df),
        columns=['PC1', 'PC2']
    ).plot(kind='scatter', x='PC1', y='PC2', figsize=(10, 8), s=0.
8)

    # Plot arrows and labels
    for i, (pc1, pc2) in enumerate(zip(pca.components_[0], pca.comp
onents_[1])):
        ax.arrow(0, 0, pc1, pc2, width=0.001, fc='orange', ec='oran
ge')
        ax.annotate(df.columns[i], (pc1, pc2), size=12)

    return ax
```

```
In [302]: ax = biplot(data)
          # Play around with the ranges for scaling the plot
          ax.set_xlim([-1.5, 1])
          ax.set_ylim([-0.25, 1])

Out[302]: (-0.25, 1)
```



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.

Here is the correlation matrix we have from our PCA

Fresh Milk Grocery Frozen Detergents_Paper Delicatessen

[-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]

Looking at PCA 0 - here is the analysis of the first component: [-0.97653685 -0.12118407 -0.06154039 -0.15236462 0.00705417 -0.06810471] From the bar graph and biploy, we see this dimension capturing high variation in customers buying Fresh and Frozen. Only Variation along Detergents_Paper seems anit-correlated.

Looking at PCA 1 - We see this dimension capturing customers high variation in buying Grocery, Milk and Detergents and Paper. Variation along Fresh and Frozen seem to be anti-correlated. [-0.11061386 0.51580216 0.76460638 -0.01872345 0.36535076 0.05707921]

Looking at PCA 2 - Captures High Variation along almost all the variables. The category seems to have correlation among Fresh, Grocery and Detergents and Paper and anti-correlated with Milk, Frozen and Delicatessen. [-0.17855726 0.50988675 -0.27578088 0.71420037 -0.20440987 0.28321747]

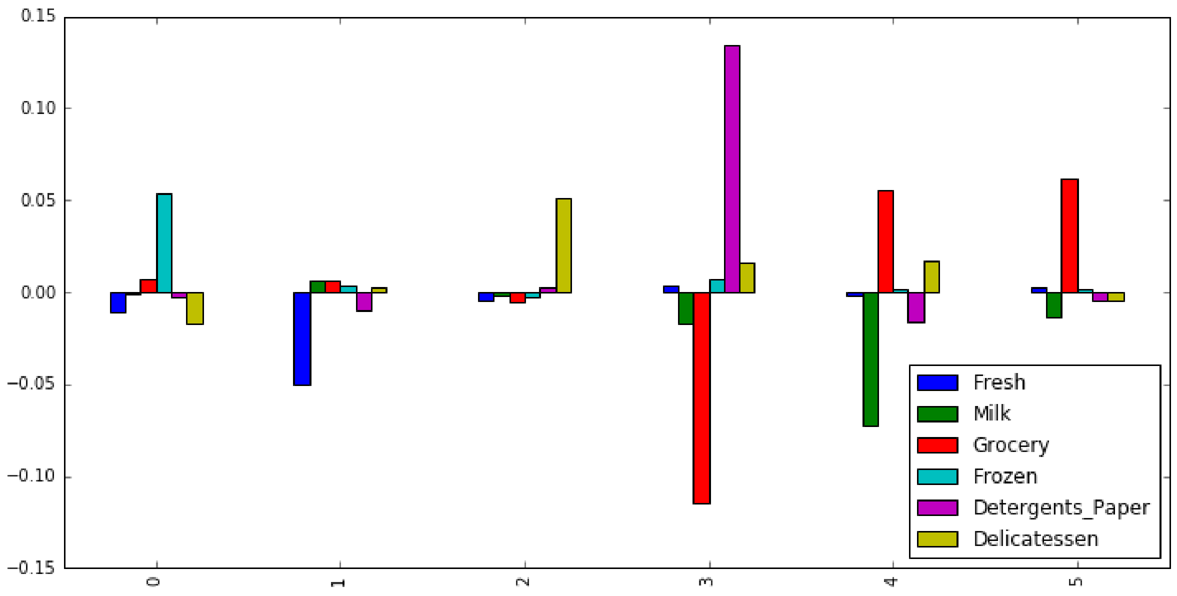
Directionality - From the feedback, I understand that "the sign of the coefficient really just determine the direction of the vector and how the features correlate to one another". I certianly can observe that in the biplot and the bar graph. From the reference, http://forrest.psych.unc.edu/research/vista-frames/help/lecturenotes/lecture13/biplot.html (http://forrest.psych.unc.edu/research/vista-frames/help/lecturenotes/lecture13/biplot.html)

It tells me the direction of the vector for a variable represents the direction in which the variable shows the most correlation to the principal component. It also indicates that "vectors that point in the same direction correspond to variables that have similar response profiles". The response profile in our context will be customer types who would show a similiar variation in spending on buying certain units of those variables in consideration.

ICA

```
In [303]: # 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 import preprocessing
          ica = FastICA(n_components=6,random_state=6)

          #for each feature, obtain the mean and standard deviation to use it
          later to transform the centroids to the orignal
          #space.

          standard_scaler = preprocessing.StandardScaler()
          standard_scaler.fit_transform(data)

          data /= data.std(axis=0)
          ica.fit(data)

          pd.DataFrame(ica.components_,columns=data.columns).plot(kind='bar',
          figsize = (12,6))

          # Print the independent components
          print ica.components_
          #print scaler.mean_
          #print scaler.scale_
```

```
[[-0.01094139 -0.00103854  0.00735399  0.05411141 -0.00263969 -0.0
1678528]
 [-0.05028679  0.00635215  0.00601695  0.00328508 -0.00991827  0.0
029289 ]
 [-0.00488842 -0.00162034 -0.00569967 -0.00253455  0.00242833  0.0
5102196]
 [ 0.00377098 -0.0171225  -0.11429953  0.00711047  0.13445226  0.0
1615758]
 [-0.00194711 -0.07265051  0.05525802  0.00176226 -0.01587578  0.0
1707609]
 [ 0.00265178 -0.01385789  0.0615885   0.00196223 -0.00444568 -0.0
0417354]]
```



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 attempts to decompose a multivariate signal into independent non-gaussian signals even though it cannot identify the actual number of source signlas. [taken from wikipedia]

what we are seeing is the unmixing matrix which decomposes the linearly mixed data into a sum of temporally independent and spatially fixed components. The columns give the relative projection strengths of the respective features at each of the component or dimension.

Directionality and Magnitude - In helping to identify the source, the magniture indicates how much of the projection of the variable we need and the direction indicates the directions of the correlation. So, as a two step process, we identify the prevelance by magnitude and then we will try to interpret the directionality.

Regarding the directionality, I found the following to be very useful explanationa:

Kurtosis is a classical measure of non-Gaussianity of a distribution. The less Gaussian the distribution, the higher the absolute value of kurtosis. While a Gaussian has a kurtosis of zero, platykurtic (sub-Gaussian) distributions with broad and wide peaks have negative kurtosis, and leptokurtic (super-Gaussian) distributions with acute and narrow peaks have positive kurtosis. Reference - http://pressrelease.brainproducts.com/independent-component-analysis-demystified/ (http://pressrelease.brainproducts.com/independent-component-analysis-demystified/)

Since ICA can only successfully unmix non-Gaussian signals, the positive and negative signals help identify the variables that could help unmix. Closer to zero, ICA find the signals less useful to identify the sources.

Here is the matrix [-0.01094139 -0.00103854 0.00735399 0.05411141 -0.00263969 -0.01678528] [-0.05028679 0.00635215 0.00601695 0.00328508 -0.00991827 0.0029289] [-0.00488842 -0.00162034 -0.00569967 -0.00253455 0.00242833 0.05102196] [0.00377098 -0.0171225 -0.11429953 0.00711047 0.13445226 0.01615758] [-0.00194711 -0.07265051 0.05525802 0.00176226 -0.01587578 0.01707609] [0.00265178 -0.01385789 0.0615885 0.00196223 -0.00444568 -0.00417354]]

Therefore, Interpretation of ICA 1:

Fresh Milk Grocery Frozen Detergents_Paper Delicatessen [-0.01094139 -0.00103854 0.00735399 0.05411141 -0.00263969 -0.01678528]

This component helps identify a source that is positively correlated on Frozen and little bit of Grocery and little anit-correlated to Delicatssen and almost zero on others. This could indicate a store that sells a lot of Frozen like a 'Dollar Store' or one that sells Delicatessen like a gas station mart on interstates.

Interpreation of ICA 3: Fresh Milk Grocery Frozen Detergents_Paper Delicatessen [-0.00488842 -0.00162034 -0.00569967 -0.00253455 0.00242833 0.05102196]

Clearly, this one sells more of Delicatessen and very less on others. Could be a Local Deli Store or Pizza Place.

Interpretation of ICA 4:

Fresh Milk Grocery Frozen Detergents_Paper Delicatessen [0.00377098 -0.0171225 -0.11429953 0.00711047 0.13445226 0.01615758]

The coefficients of Grocery and Detergents are way higher than others and also that these are anti-correlated. Detergents is correlated with Delicatessen while we see correlation between [Milk and Grocery]. So, this category could represent a grocery store "or" a high volume whole sale store that has a instore Delicatessen like Costco or Sams Club.

Interpreation of ICA 6: Fresh Milk Grocery Frozen Detergents_Paper Delicatessen [0.00265178 -0.01385789 0.0615885 0.00196223 -0.00444568 -0.00417354]]

Grocery coefficient is higher than the rest followed by Milk. But, Grocery and Milk are 'anti-correlated'. So, it could indicate a grocery store that sells Grocery or a convenient that sells Milk related products.

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 as it uses hard clustering - that is, our case, each customer will be assigned to only one cluster based on the eucladian distance. 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 models.
        b. It can also draw confidence ellipsoids for multivariate models
(Confidence Ellipsoid - defines the region where certain percent of all
the samples that could be drawn from the underlying Gaussian distributi
on),
        c. and compute the Bayesian Information Criterion to assess the nu
mber of clusters in the data (Bayesian  Information Criterior - BIC - i
s a criterion for model selection  among a finite set of models; the mo
del with the lowest BIC is preferred. BIC limits the parameters used in
the model by penalizing while trying to increase the likelihood).
        d. uses soft clustering - a probability score - for assigning obje
cts to clusters. Objects,
         customers in our case, could be assigned to multiple clusters of
buyer or spending types.
```

inference: So, computationally, GMM is doing more that a straight forward eucladian based k-means clustering and hence slower than k-means.

Choice of use: i make a choice to use gmm becase a. we can force the model to use bic to calculate the number of components and avoid overfitting. b. starting with n-of clusters on k-means as a parameter assumes that you have prior knowledge into the situation and can lead to less optimized model more certainly than not. c. provide confidence on the ellipsoids. d. uses expectaton maxmimization - well established algorithm - to overcome the difficult of knowing which point came from which component and gaurantees converging to local optimium.

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 [304]:
```
# Import clustering modules
from sklearn.cluster import KMeans
from sklearn.mixture import GMM
```

In [305]:
```
# TODO: First we reduce the data to two dimensions using PCA to cap
ture variation
pca = PCA(n_components=2);
reduced_data = pca.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 [306]:
```
# TODO: Implement your clustering algorithm here, and fit it to the
reduced data for visualization
# The visualizer below assumes your clustering object is named 'clu
sters'

kmeans_clusters = KMeans(init='k-means++', n_clusters=4, n_init=10)
#?
kmeans_clusters.fit(reduced_data)
print kmeans_clusters
```

```
KMeans(copy_x=True, init='k-means++', max_iter=300, n_clusters=4,
n_init=10,
    n_jobs=1, precompute_distances='auto', random_state=None, tol=
0.0001,
    verbose=0)
```

In [307]:
```
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_typ
e)
        gmm.fit(reduced_data)
        bic.append(gmm.bic(reduced_data))
        if bic[-1] < lowest_bic:
            lowest_bic = bic[-1]
            best_gmm = gmm
gmm_clusters=best_gmm
print gmm_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
=None,
  thresh=None, tol=0.001, verbose=0)
```

In [308]:
```
# 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 = gmm_clusters.predict(np.c_[xx.ravel(), yy.ravel()])
print Z
```

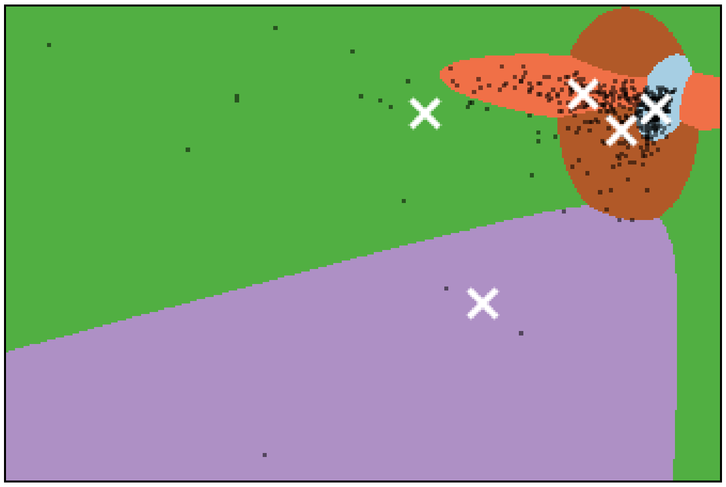
```
[3 3 3 ..., 1 1 1]
```

In [309]:
```
# TODO: Find the centroids for KMeans or the cluster means for GMM

#centroids = kmeans_clusters.cluster_centers_
centroids = gmm_clusters.means_
print centroids
```

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

```
In [310]: # 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()
```

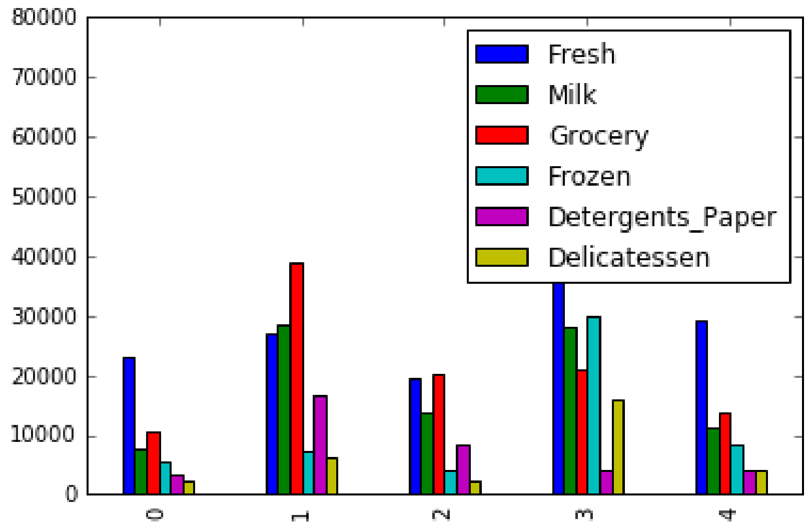
Clustering on the wholesale grocery dataset (PCA-reduced data)
Centroids are marked with white cross



In [311]:
```
# Bringing the centroid back to Original Space.
centers = pca.inverse_transform(centroids)
#print(centers)
#using scikit StandardScalar(created in step 303) object to transfo
rm back to original space
centers_os = standard_scaler.inverse_transform(centers)
print(centers_os) #centers transformed to original space


pd.DataFrame(centers_os,columns=data.columns).plot(kind="Bar")
```

```
[[ 22946.29064744   7598.61438181  10607.80561961   5670.55443858
    3290.4829041    2252.6208022 ]
 [ 26910.76840666  28594.75212628  38952.88211148   7471.24258248
   16664.09481885   6138.73139707]
 [ 19393.10016407  13642.05809639  20352.72917749   4103.28656631
    8290.30348858   2435.1095862 ]
 [ 77148.6183185   27957.6395005   21076.92819111  29789.53533595
    3917.23053946  16062.52831144]
 [ 29056.99372473  11210.2824566   13649.25174698   8392.16645801
    4260.55577659   4003.61880804]]
```

Out[311]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11b43fe10>
```



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

Answer: There are five clusters are they are visible clearly. The centroids basically denotes the 'average' customer in each cluster. What I mean by average is that the PCA fit data basically identifies a customer model (a vector) whose feature values are the mean of each feature for customers represented in that cluster. The same is applicable for the set of clusters the model produced.

```
    Fresh             Milk          Grocery          Frozen      Detergen
  ts_Paper  Delicatessen
```

[[22928.21953454 7594.63474082 10608.28530854 5662.51840977 3292.19250688 2248.4305743] [19376.9913684 13663.33158361 20388.24906402 4096.16969433 8308.78586314 2435.0338563] [28943.94489312 11187.4666458 13655.19321191 8341.89876261 4272.67173759 3977.71233665] [27037.94998305 28550.15382627 38846.85414351 7527.66273064 16602.4561356 6157.51680966] [76304.46385793 27501.76531865 20717.64317112 29413.64280178 3812.63734762 15826.98692632]]

Looking at the first centroid, [19376.9913684 13663.33158361 20388.24906402 4096.16969433 8308.78586314 2435.0338563] It seems the clusters are having customers that have more spending on Groceries, Fresh and Milk followed by relatively (50%) less on Detergents and Paper, Frozen and Delicatessen. The particular cluster is trying to pull those who have the spending habits of heavy on groceries, fresh and milk which could tell it is a local grocery store.

The last cluster [76304.46385793 27501.76531865 20717.64317112 29413.64280178 3812.63734762 15826.98692632]] pulls customers who are overall spend-heavy especially on Fresh, followed by heavy spending (Frozen, Milk, Grocery, Delicatessen) on all others and relatively less on detergents and paper. It could actually represent high volume customers like Super Walmart who intend to sell everything and usually put the detergents and paper section to the back of the store.

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)

PCA and ICA helps us to look at more than one dimension at a time. PCA helps us cluster the customers through maximing the variation of their spending while ICA working through maximizing independents on their spending habits. Both the methods helps us separate the signal from the noise. Even though ICA cannot exactly pin point the number of independent signals, research into maximixing the parameters of ICA and Clustering techniques could improve the overall process.

The thing that I learnt the most from this project is about the unmxing matrix in ICA and covariance matrix in PCA and helped look at the problem holistically and not let one dimesion drive the analysis.

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.