

customer_segments

February 2, 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 [173]: # 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

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:

For PCA dimensions, my guess is that a combined component of “Fresh”, “Milk”, and “Grocery” will be the first principal component. In my opinion, “Fresh”, “Milk”, and “Grocery” all fall under the more general category of “groceries”.

For ICA dimensions, my guess is that “Detergents_Paper” will be a basis vector of ICA. Out of all product categories (features) in our data, “Detergents_Paper” seems like the most different/independent category compared to the rest.

1.1.1 PCA

```
In [174]: # TODO: Apply PCA with the same number of dimensions as variables in the dataset
from sklearn.decomposition import PCA

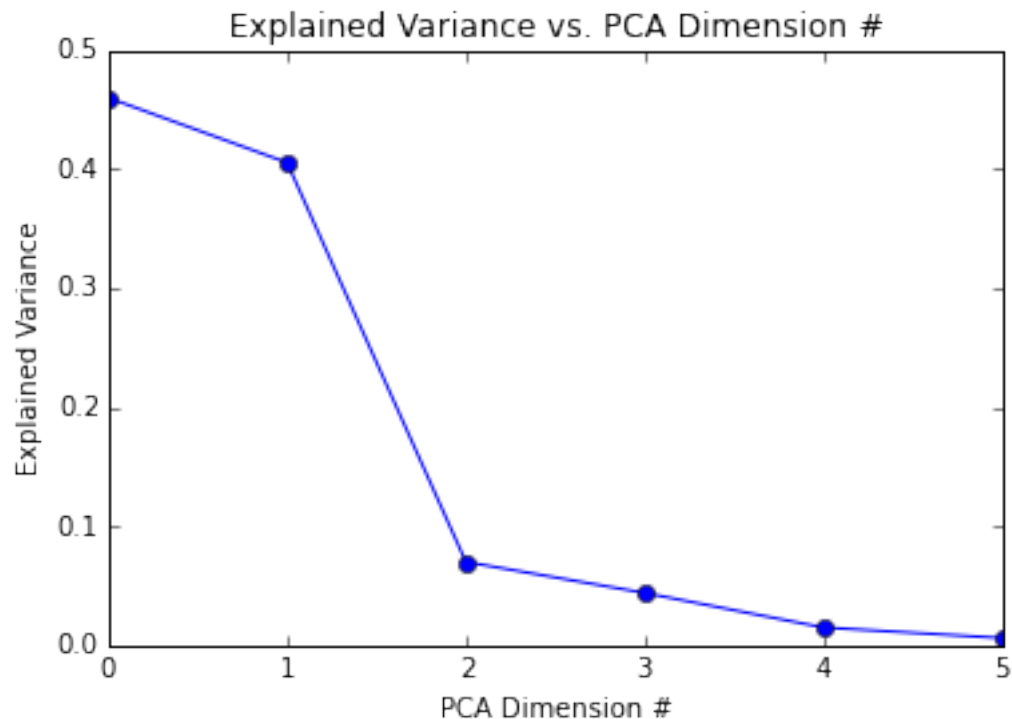
pca = PCA(n_components=len(data.columns))
pca.fit(data)

# 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]]

In [175]: # Plot the explained variance as function of PCA dimension number
plt.plot(pca.explained_variance_ratio_, 'bo-')
plt.xlabel('PCA Dimension #')
plt.ylabel('Explained Variance')
plt.title('Explained Variance vs. PCA Dimension #')
```

Out[175]: <matplotlib.text.Text at 0x254ea710>



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:

Looking at PCA's explained variance ratio, we see that the variance drops off significantly after the second principle component/dimension. In particular, the first to second principle components explain 46% and 41% of the total variance, respectively. However, the third principle component only explains 7% of the total variance. From the above plot, we can clearly see the sharp drop-off in explained variance after the second principle component.

Based on the above reasoning, if I were to use PCA on this dataset, I would choose 2 dimensions for my analysis.

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

Answer:

The first dimension (first principle component) is basically the "Fresh" feature, but negative. Looking at the first principle component, we see the first value (the value corresponding to "Fresh") is -0.9765, whereas all other values are much closer to 0. The fact that this value is negative is acceptable, as this means we found the appropriate eigenvector, just pointing in the opposite direction.

The second dimension (second principle component) seems to represent a weighted combination of "Grocery", "Milk", and "Detergents_Paper", since the values corresponding to the aforementioned features in the second principle component seem to be the largest.

From the analysis above, it seems we can determine our customer segments using just these two dimensions. Using the two principle components described above, we can run unsupervised clustering algorithms to discover customer segments.

Note that the "Frozen" and "Delicatessen" features do not play a large role in our reduced feature set, because the volume of sales in those categories are too low to create meaningful customer segments. If there is a particular category of customers that focus on purchasing "Frozen" or "Delicatessen" products, their volume is too low for us to allocate resources to target them, relative to other customer categories.

1.1.2 ICA

```
In [176]: # TODO: Fit an ICA model to the data
# Note: Adjust the data to have center at the origin first!
from sklearn.decomposition import FastICA

# Adjust the data to have center at the origin first
data_norm = data.apply(lambda x: (x - np.mean(x)))

# Since ICA gives different results each time, I will run ICA 3 times
# and visually inspect all components_ matrices,
# to see if results are generally similar or not
for i in range(3):
    ica = FastICA()
    ica.fit(data_norm)
    comp = ica.components_

    print 'Random Polarity ICA Run ' + str(i)
    print comp[np.argsort(comp[:,0])]
    print '\n'

# From the results in the above loop, we will see the ICA
# components' polarity is random. Let's fix the polarity
# to try to spot some patterns
for i in range(3):
    ica = FastICA()
    ica.fit(data_norm)
    comp = ica.components_
```

```

# ICA gives arbitrary polarity for components,
# so standardize the polarity such that
# the first column's value is always positive
for row in comp:
    if row[0] < 0:
        row *= -1

print 'Fixed Polarity ICA Run ' + str(i)
print comp[np.argsort(comp[:,0])]
print '\n'

```

Random Polarity ICA Run 0

```

[[ -8.65113651e-07  -1.40718982e-07   7.73844658e-07   1.11462531e-05
  -5.53630939e-07  -5.95200027e-06]
 [ -3.86518605e-07  -2.19546787e-07  -5.99756823e-07  -5.22083890e-07
   5.09306013e-07   1.80922094e-05]
 [  1.53956780e-07   9.84372428e-06  -5.81474193e-06  -3.62995169e-07
   3.32988486e-06  -6.05510118e-06]
 [  2.09663440e-07  -1.87758757e-06   6.48136789e-06   4.04127232e-07
  -9.33639087e-07  -1.48017401e-06]
 [  2.98149460e-07  -2.32012533e-06  -1.20272178e-05   1.46468274e-06
   2.81996730e-05   5.72940256e-06]
 [  3.97608150e-06  -8.60695137e-07  -6.33227683e-07  -6.76676954e-07
   2.08039983e-06  -1.03853756e-06]]

```

Random Polarity ICA Run 1

```

[[ -3.01238198e-07   2.29366198e-06   1.20947880e-05  -1.46073243e-06
  -2.82130281e-05  -5.73448364e-06]
 [  1.53345150e-07   9.84678371e-06  -5.80564718e-06  -3.64623204e-07
   3.30272020e-06  -6.05956978e-06]
 [  2.11404984e-07  -1.89539682e-06   6.36407050e-06   4.19182441e-07
  -6.63748165e-07  -1.42440243e-06]
 [  3.86370125e-07   2.19517024e-07   6.01695502e-07   5.22034622e-07
  -5.10988443e-07  -1.80928701e-05]
 [  8.65277867e-07   1.40191593e-07  -7.74392699e-07  -1.11460785e-05
   5.56513828e-07   5.95258509e-06]
 [  3.97575880e-06  -8.57681750e-07  -6.16677595e-07  -6.78094408e-07
   2.04476080e-06  -1.04751891e-06]]

```

Random Polarity ICA Run 2

```

[[ -2.58046865e-07   2.64265425e-06   1.13034531e-05  -1.50267549e-06
  -2.79160642e-05  -5.70043576e-06]
 [ -1.95715336e-07   1.75986825e-06  -7.54728164e-06  -2.60012895e-07
   3.39324355e-06   1.94373589e-06]
 [  1.70027969e-07   9.77860717e-06  -5.96049701e-06  -3.40194272e-07
   3.82303065e-06  -5.93449960e-06]
 [  3.88885133e-07   2.41423466e-07   5.79355296e-07   5.25419848e-07
  -5.06955617e-07  -1.81058643e-05]
 [  8.62571270e-07   1.41645771e-07  -7.75316104e-07  -1.11469059e-05
   5.37491397e-07   5.94533227e-06]
 [  3.97926299e-06  -9.14198212e-07  -7.99398542e-07  -6.62458769e-07
   2.46842696e-06  -9.23832858e-07]]

```

Fixed Polarity ICA Run 0

```
[[ 1.46306013e-07  9.82489042e-06 -5.93249610e-06 -3.45263778e-07
  3.59616956e-06 -6.02696767e-06]
 [ 1.93944749e-07 -1.72159684e-06  7.18598696e-06  3.08869882e-07
 -2.62423089e-06 -1.85436868e-06]
 [ 2.81123785e-07 -2.51730410e-06 -1.15547292e-05  1.48686703e-06
  2.80440984e-05  5.70305706e-06]
 [ 3.89887801e-07  2.04826306e-07  5.94289064e-07  4.94601371e-07
 -5.03077893e-07 -1.80635580e-05]
 [ 8.64154837e-07  1.43824159e-07 -7.72324627e-07 -1.11487865e-05
  5.37962943e-07  5.99335216e-06]
 [ 3.97828735e-06 -8.50997381e-07 -7.33819841e-07 -6.66595458e-07
  2.28962206e-06 -1.00689922e-06]]
```

Fixed Polarity ICA Run 1

```
[[ 1.62875020e-07  9.81858954e-06 -5.85759777e-06 -3.67839931e-07
  3.49371280e-06 -5.93421503e-06]
 [ 2.01422095e-07 -1.79669936e-06  7.13521136e-06  3.17012564e-07
 -2.44075907e-06 -1.76250285e-06]
 [ 2.74546577e-07 -2.46759943e-06 -1.16256546e-05  1.48673165e-06
  2.80714010e-05  5.68681921e-06]
 [ 3.89023454e-07  2.69505089e-07  5.68801862e-07  5.29211202e-07
 -4.94535763e-07 -1.81281522e-05]
 [ 8.64061677e-07  1.33882251e-07 -7.67695215e-07 -1.11462448e-05
  5.37844044e-07  5.94110224e-06]
 [ 3.97783611e-06 -8.96431612e-07 -7.35807909e-07 -6.66924111e-07
  2.31831953e-06 -9.62440229e-07]]
```

Fixed Polarity ICA Run 2

```
[[ 1.53704476e-07  9.84502591e-06 -5.81118503e-06 -3.63665449e-07
  3.31878580e-06 -6.05689781e-06]
 [ 2.10172123e-07 -1.88170145e-06  6.44887241e-06  4.08278125e-07
 -8.59531071e-07 -1.46544342e-06]
 [ 2.98964815e-07 -2.31158905e-06 -1.20465865e-05  1.46355228e-06
  2.82040079e-05  5.73019881e-06]
 [ 3.86482975e-07  2.19551846e-07  6.00187376e-07  5.22101824e-07
 -5.09559550e-07 -1.80923830e-05]
 [ 8.65157920e-07  1.40568001e-07 -7.73931744e-07 -1.11462044e-05
  5.54356253e-07  5.95210460e-06]
 [ 3.97599702e-06 -8.59814731e-07 -6.28945238e-07 -6.77058481e-07
  2.07103374e-06 -1.04093272e-06]]
```

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:

First, looking at results from “Fixed Polarity ICA Run [0-2]”, we see that the last 4 vectors/rows of the unmixing matrices are consistent (rows 2-5, inclusive). Note all rows are zero-mean.

However, we cannot make any conclusions about the qualitative type of customer based on these 4 vectors, because their polarity is indeterminant. Looking at “Random Polarity ICA Run [0-2]”, we can find the last 4 vectors from “Fixed Polarity ICA Run [0-2]”, but with random polarity.

For example, take the vector [8.63287386e-07 1.46228026e-07 -7.75755058e-07 -1.11474875e-05 5.37380302e-07 5.95000338e-06] from “Fixed Polarity ICA Run 2”. The value that multiplies the “Fresh” feature is 8.63287386e-07, and the value that multiplies the “Grocery” feature is -7.75755058e-07. The implication is that this vector will heavily weigh the “Fresh” feature and put low weight on the “Grocery” feature. However, if the polarity is flipped, the implications on “Fresh” and “Grocery” will be flipped. With indeterminant polarity, we have indeterminant conclusions about what the vectors mean.

Unfortunately, I cannot draw any conclusions to the problem at hand, based on ICA. ICA is more applicable to problems like blind source separation (e.g. http://scikit-learn.org/stable/auto_examples/decomposition/plot_ica_blind_source_separation.html). But for our particular case of customer segment discovery, it looks like ICA is not a good method to use.

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:

The advantages of K-Means clustering is that it is a fast way to perform clustering (in terms of computational runtime), compared to Gaussian Mixture Models. However, K-Means clustering only provides information about which cluster a data point belongs to, but no information about the likelihood/probability of that data point belonging to said cluster.

This is in contrast to clustering using Gaussian Mixture Models. Although it runs slower than K-Means, Gaussian Mixture Models can not only provide information about which cluster a data point belongs to, it can also provide the probability that a particular data point belongs to said cluster. Also, Gaussian Mixture Models can calculate more complex decision boundaries compared to K-Means.

Since we have a small dataset for the computational power available on standard personal computers, I believe Gaussian Mixture Models will work better for our problem at hand.

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

In [178]: # TODO: First we reduce the data to two dimensions using PCA to capture variation
          pca = PCA(n_components=2)
          reduced_data = pca.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]]
```

Using K-Means Clustering

This is included for completeness, to contrast the decision boundaries of K-Means versus Gaussian Mixture Models

```
In [179]: # TODO: Implement your clustering algorithm here, and fit it to the reduced data for visualiz
# The visualizer below assumes your clustering object is named 'clusters'
num_clusters = 3
clusters = KMeans(n_clusters=num_clusters).fit(reduced_data)

print clusters

KMeans(copy_x=True, init='k-means++', max_iter=300, n_clusters=3, n_init=10,
       n_jobs=1, precompute_distances='auto', random_state=None, tol=0.0001,
       verbose=0)

In [180]: # 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 [181]: # TODO: Find the centroids for KMeans or the cluster means for GMM
centroids = clusters.cluster_centers_
print centroids

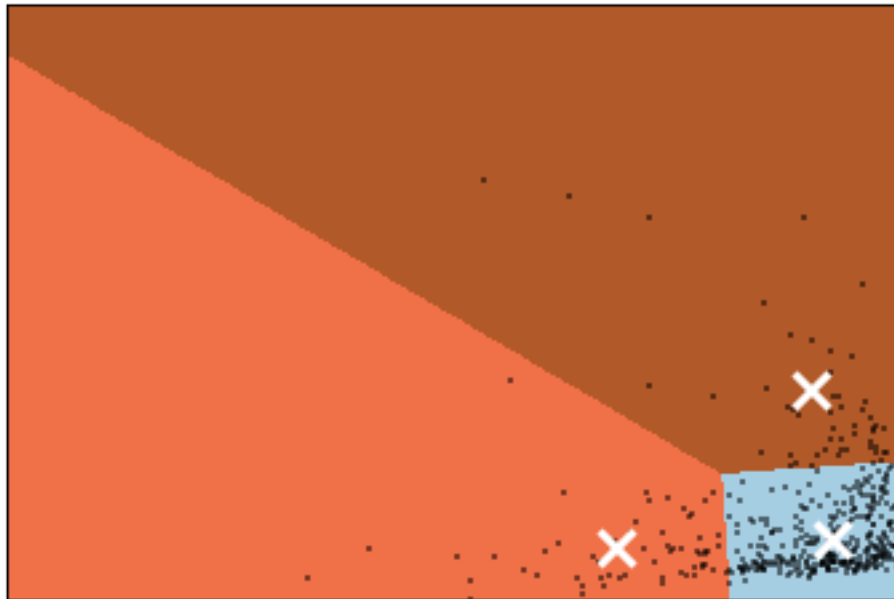
[[ 4165.1217824  -3105.15811456]
 [-23978.86566553 -4445.56611772]
 [ 1341.31124554  25261.39189714]]

In [182]: # 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



Using Gaussian Mixture Model Clustering

First, try clustering using 2 clusters, to visually see what kind of decision boundary we get

```
In [183]: # TODO: Implement your clustering algorithm here, and fit it to the reduced data for visualiz
# The visualizer below assumes your clustering object is named 'clusters'
clusters = GMM(n_components=2).fit(reduced_data)
```

```
print clusters
```

```
GMM(covariance_type='diag', init_params='wmc', min_covar=0.001,
    n_components=2, n_init=1, n_iter=100, params='wmc', random_state=None,
    thresh=None, tol=0.001)
```

```
In [184]: # 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 [185]: # TODO: Find the centroids for KMeans or the cluster means for GMM
centroids = clusters.means_
print centroids
```

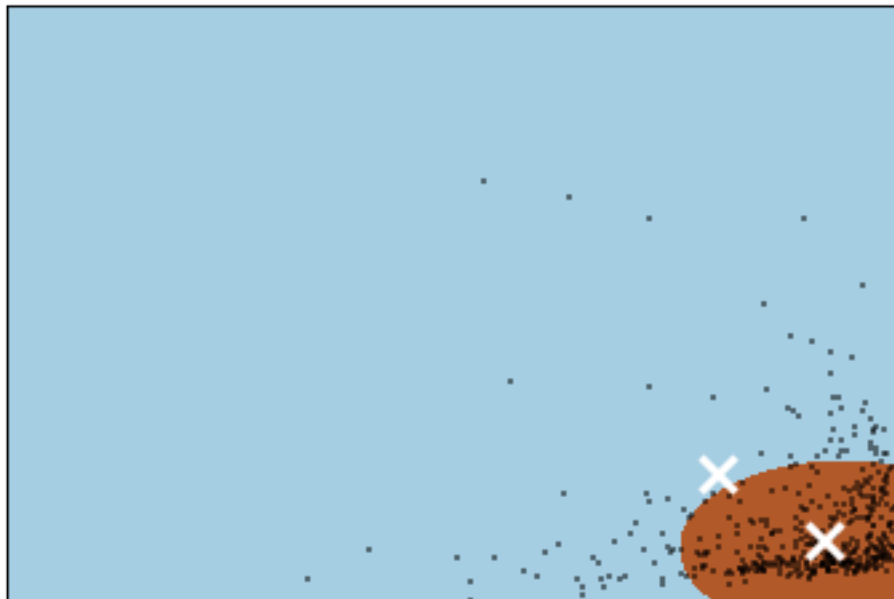
```
[[ -10810.23008886   9858.15532401]
 [  3308.39301792  -3017.01739698]]
```



```
In [186]: # 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



Using 2 clusters above, the clustering and decision boundary does not look right. Let's use 3 clusters and re-run our clustering algorithm

```
In [202]: # TODO: Implement your clustering algorithm here, and fit it to the reduced data for visualiz
# The visualizer below assumes your clustering object is named 'clusters'
clusters = GMM(n_components=num_clusters).fit(reduced_data)

print clusters
```

```
GMM(covariance_type='diag', init_params='wmc', min_covar=0.001,
    n_components=3, n_init=1, n_iter=100, params='wmc', random_state=None,
    thresh=None, tol=0.001)
```

```
In [203]: # 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()])
```

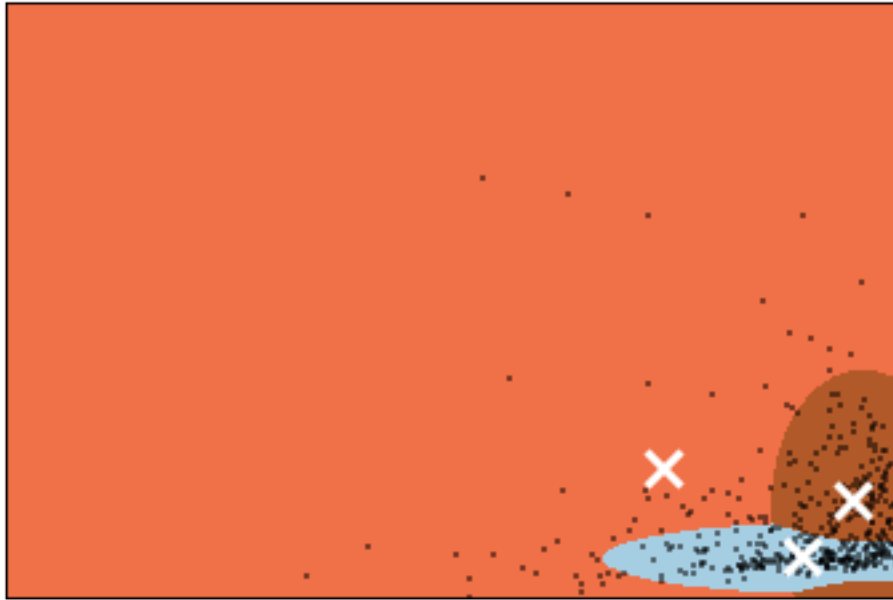
```
In [204]: # TODO: Find the centroids for KMeans or the cluster means for GMM
centroids = clusters.means_
print centroids
```

```
[[ 269.05318679 -6506.88683442]
 [-17879.18623839 10122.79246625]
 [ 6987.95079141  4249.82914044]]
```

```
In [205]: # 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



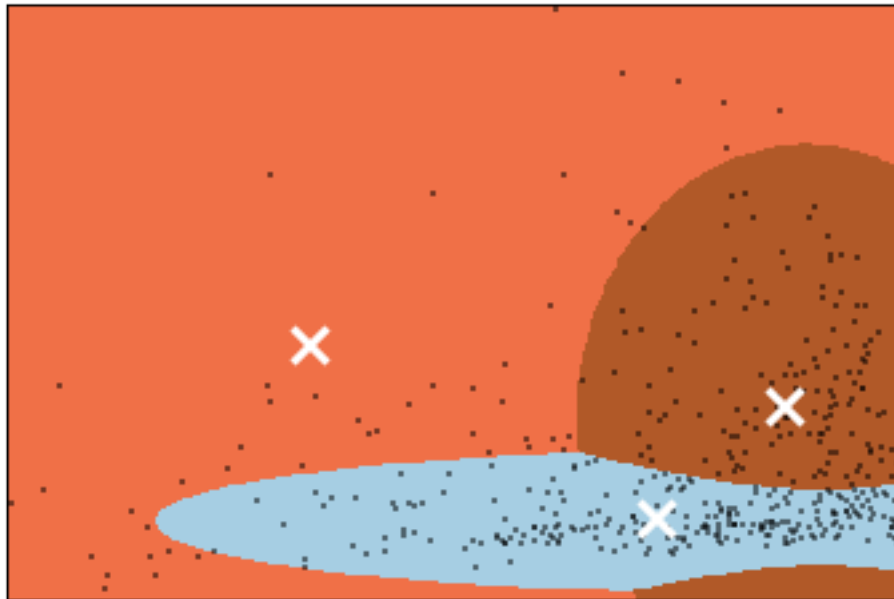
Zoom in on the plot above

```
In [206]: # Same plot as above, but zoomed-in
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 + 0.6*(x_max-x_min), x_max)
plt.ylim(y_min, y_max - 0.5*(y_max-y_min))
plt.xticks(())
plt.yticks(())
plt.show()
```

Clustering on the wholesale grocery dataset (PCA-reduced data)

Centroids are marked with white cross



Sampling elements from each cluster and interpreting them Let's just take a look at the centroid for each of the 3 clusters:

centroid of blue cluster = [313.48539412 -6497.20521133] centroid of brown cluster = [7007.45427668 4294.01360677] centroid of orange cluster = [-17858.6536729 10050.33930164]

Looking at the centroid of the blue cluster, this looks like a customer who buys little/moderate amounts of fresh produce, and very little amounts of groceries/milk/detergents/paper.

Looking at the centroid of the brown cluster, this looks like a customer who buys moderate amounts of groceries/milk/detergents/paper, and very little fresh produce.

Looking at the centroid of the orange cluster, this looks like a customer who buys large amounts of fresh produce, and moderate amounts of groceries/milk/detergents/paper.

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

Answer:

First, note that the horizontal axis is our first principle component from PCA, which is the component dominated by the "Fresh" category. Also note the horizontal axis is negative, so datapoints on the left of the plot represent customers who placed relatively large orders of "Fresh" items. The vertical axis is our second principle component from PCA, which is a combination of "Grocery", "Milk", and "Detergents.Paper". The datapoints on the top of the plot represent customers who placed large orders of groceries/milk/detergents/paper.

Recall that "Frozen" and "Delicatessen" categories are not represented in our 2 chosen principal components, since the quantities of frozen and delicatessen products are too small compared to the other categories. This renders frozen and delicatessen categories less relevant.

Qualitatively, the blue cluster represents customers who buy very little groceries/milk/detergents/paper, and little to moderate quantities of fresh produce. The customers represented by the blue cluster appear to be restaurants.

The brown cluster represents customers who buy very little fresh produce, but moderate to large quantities of groceries/milk/detergents/paper. The customers represented by the brown cluster appear to be supermarkets.

The orange cluster represents customers not included in the blue and brown clusters, and it's hard to pinpoint an exact category of customers for the orange cluster. However, the customers in the orange category are large customers, so they are an important category to consider. This category can be referred to as "other large customers".

1.2.2 Conclusions

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

Answer:

For deciding between PCA and ICA, I felt PCA worked better for the problem at hand. For more details, see answers to questions 3 and 4.

Deciding between K-Means and Gaussian Mixture Model (GMM) clustering, I felt GMM gave more insight into the data. Looking at the plots for both K-Means and GMM clustering methods (question 6), the plot using GMM clustering made more intuitive sense, in terms of valid clustering decision boundaries. Also, computational runtime was not an issue for GMM, given the small size of our dataset and available computing power in standard personal computers. Recall one disadvantage for GMM versus K-Means is that GMM requires more computation time.

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

Answer:

As seen from our analysis using PCA and GMM clustering, we can find 3 general customer segments: restaurants, supermarkets, and "other large customers". This information can help the company design an A/B test to better evaluate the effects of any new policies.

From <https://discussions.udacity.com/t/improving-the-ability-to-get-good-results-from-an-a-b-test/44142/2>, "[i]n an A/B test, you split your customers into two sets, and implement a small change on one set of customers. The idea is that if your two sets of customers are very similar otherwise, the small change should account for any differences in their behavior. You can then check if there are significant changes in behavior, before deciding whether to implement the change for everyone."

For example, to test the effects of changing from regular morning delivery to bulk evening delivery, the company can run A/B tests on its 3 main customer segments separately. Starting with its restaurant customers (as determined by our clustering algorithm), the company can choose a subset of its restaurant customers to implement the evening delivery change, and evaluate the effect versus the control group which still receives morning deliveries. The company can then repeat this for supermarkets and the other large customers, to get a more precise idea of the effects of the policy change.

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

Answer:

The feature reduction and clustering work we have done can be used to aid in supervised learning tasks in the future.

For our PCA feature reduction, we have greatly reduced the feature set from 6 features down to 2 features. This allows future supervised learning algorithm applications to run much faster. Also, we would need less data to avoid the curse of dimensionality, since we have less dimensions.

For our clustering work, we discussed above how this helps with A/B testing. Thus, any future supervised learning applications based on A/B testing results will benefit from the clustering work we have performed.