

customer_segments

February 17, 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 [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
```

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:

PCA

tries to find correlations by looking for the maximum variance. So the first dimensions for the PCs are, what has the largest range of values in the data set.

Possible first dimensions: - the amount in which a certain category is replenished - the size of the customers business

ICA

tries to transform the feature space towards maximal independence. So the first dimension will contain the most common information in the data set. And the following dimensions will consist of the least common information. Possible type of vectors dimensions: - the average replenishment profile of a category - different replenishment profiles for certain customer groups

source: <https://www.udacity.com/course/viewer#!/c-ud727-nd/l-5453051650/m-661438547>

1.1.1 PCA

```
In [2]: # TODO: Apply PCA with the same number of dimensions as variables in the dataset
        from sklearn.decomposition import PCA
```

```
pca = PCA(n_components = data.shape[1])
pca.fit(data)
#http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#sklearn.decomp

# 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]
```

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?

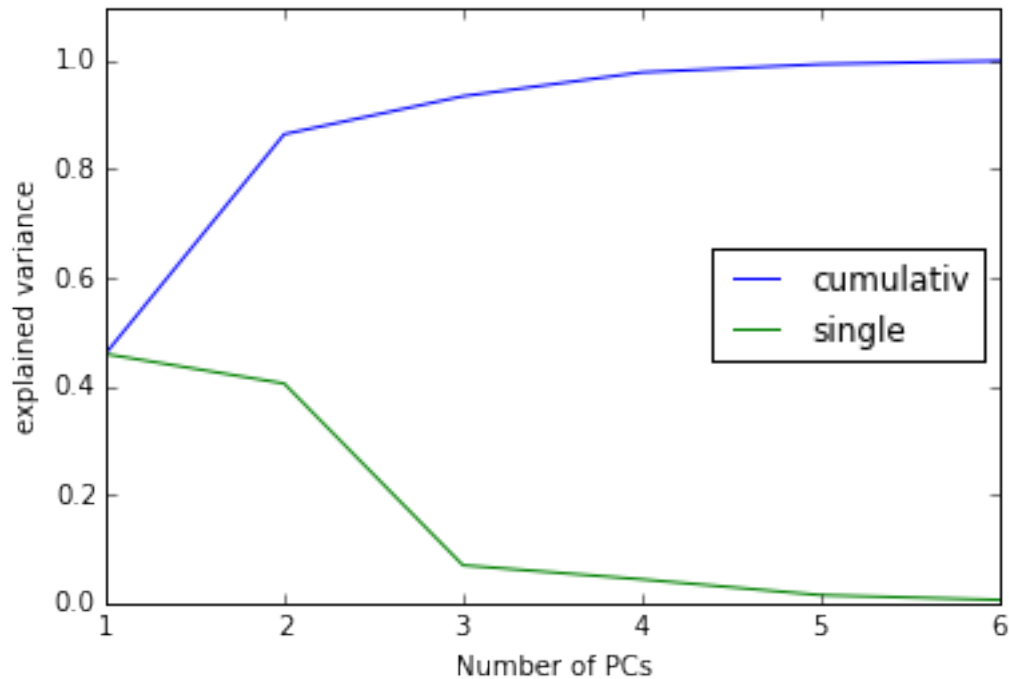
```
In [3]: # PCs variance plot
```

```
cs = np.cumsum(pca.explained_variance_ratio_)
#https://github.com/numpy/numpy/blob/v1.10.0/numpy/core/fromnumeric.py#L2038-L2106
#http://docs.scipy.org/doc/numpy-1.10.0/reference/generated/numpy.cumsum.html

# Number array for PCs
num_PCs = np.arange(data.shape[1])+1
#http://docs.scipy.org/doc/numpy-1.10.1/user/basics.creation.html

plt.plot(num_PCs, cs)
plt.plot(num_PCs, pca.explained_variance_ratio_)
#http://matplotlib.org/users/pyplot_tutorial.html

plt.axis([1, data.shape[1], 0, 1.1])
plt.xlabel('Number of PCs')
plt.ylabel('explained variance')
plt.legend(['cumulativ', 'single'], loc='center right')
plt.show()
```



Answer:

[0.45961362 0.40517227 0.07003008 0.04402344 0.01502212 0.00613848]

The first two PCs make for more than 80% of explained variance. The third PC adds 7% to explained variance. All additional PCs add less than 5% to explained variance. Depending on how much I want/need to reduce my data set, I would pick the first 2 to 3 PCs. In this case, with a relatively small dataset, it is reasonable to use 3 PCs. To retain as much information as possible. With larger datasets, I probably go for only 2 PCs, because it is computational cheaper/faster.

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

Answer:

These dimensions represent the directions in R6 that show the most variance i.e. the largest range of values.

[-0.97653685 -0.12118407 -0.06154039 -0.15236462 0.00705417 -0.06810471]

The **first dimension** of the PCA has a huge emphasis on the 'Fresh' feature and by similarity some emphasis on 'Milk' and 'Frozen'. This component captures the extent to which a customer orders **perishables** i.e. temperature controlled goods. This component has the greatest influence on customers ordering habits.

[-0.11061386 0.51580216 0.76460638 -0.01872345 0.36535076 0.05707921]

The **second dimension** of the PCA has a huge emphasis on the 'Grocery' feature and some emphasis on 'Milk' and 'Detergents Paper'. This component captures the extent to which customer orders **fast moving consumer goods**. This component also has a great influence on customers ordering habits.

[-0.17855726 0.50988675 -0.27578088 0.71420037 -0.20440987 0.28321747]

The **third dimension** of the PCA has an emphasis on the 'Frozen' feature but is otherwise not as distinct as the first two dimensions.

1.1.2 ICA

```
In [4]: # 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
```

```

# feature centering
scaler = preprocessing.StandardScaler(copy=True, with_mean=True, with_std=True).fit(data.astype(float))
data_centered = scaler.transform(data)
# http://scikit-learn.org/stable/modules/preprocessing.html#auto_examples/decomposition/pl

ica = FastICA()
ica.fit(data_centered).transform(data_centered)
# http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.FastICA.html#sklearn.d

# Print the independent components
print ica.components_

[[-0.00251976  0.01317329 -0.0687951  -0.00146515  0.01284917  0.00512613]
 [-0.00207125 -0.07226348  0.05612073  0.00161492 -0.01739495  0.01690194]
 [-0.05026019  0.00660952  0.00710853  0.00324596 -0.01117553  0.00270222]
 [ 0.00490065  0.00165661  0.00558927  0.00254135 -0.00240962 -0.05096911]
 [-0.00343196  0.018732   0.10940523 -0.00722317 -0.13346734 -0.01608606]
 [ 0.0108979   0.00114423 -0.00739184 -0.05405578  0.00256539  0.01673036]]

/Users/philipp/anaconda/lib/python2.7/site-packages/sklearn/utils/validation.py:498: UserWarning: StandardScaler got %s" % (estimator, X.dtype))

```

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:

[-0.00490526 -0.00159944 -0.00558149 -0.00251253 0.00240565 **0.05094019**]

The first component captures customers with a high preference for the 'Delicatessen' feature. These customers probably run specialized 'fine food' stores like Dean and DeLuca.

[-0.00333902 0.01923852 **0.10788913** -0.00726281 **-0.1331113** -0.0160667]

The second component captures customers with a high preference for the 'Detergents Paper' and the 'Grocery' feature. These customers probably run small corner stores like the seven eleven chain.

[0.00247052 **-0.01275896** **0.07072851** 0.00132303 **-0.01526677** -0.00546721]

The third component captures customers with a high preference for the 'Grocery' feature and some preference for 'Detergents Paper' and 'Milk'. These customers probably run kiosk-style stores like gas stations.

[0.00205215 **0.07220843** **-0.05661183** -0.00163037 **0.01792716** **-0.01686188**]

The fourth component captures customers with a broad set of preferences like the 'Milk', 'Grocery', 'Detergents Paper', and 'Delicatessen' feature. These customers probably run supermarkets.

[**-0.05026781** 0.00659388 0.00744547 0.00321647 **-0.01153712** 0.00267678]

The fifth component captures customers with a high preference for the 'Fresh' feature and some preference for 'Detergents Paper'. These customers probably run fruit and vegetable stores.

[**-0.01090446** -0.00108012 0.00735809 **0.05405678** -0.00255502 **-0.01677352**] *The sixth component captures customers with a high preference for the 'Frozen' and some preference for 'Fresh' and 'Delicatessen'. These customers probably run central food markets or upper-end supermarket chains.*

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:

K-Means advantages

- “scales well to [a] large number of samples.”
- “General-purpose” clustering algorithm
- “has been used (successfully) across a large range of application areas.”
- “K-Means can be seen as a special case of Gaussian mixture model with equal covariance per component.”

Gaussian Mixture Models advantages

- “incorporate information about the covariance structure of the data.”
- “can also draw confidence ellipsoids for multivariate models.”
- “compute the Bayesian Information Criterion to assess the number of clusters in the data.”
- “it is the fastest algorithm for learning mixture models.”
- “as this algorithm maximizes only the likelihood, it will not be bias the means towards zero, or bias the cluster sizes to have specific structures that might or might not apply.”

source: <http://scikit-learn.org/stable/modules/mixture.html#mixture>, <http://scikit-learn.org/stable/modules/clustering.html#k-means>, <https://www.udacity.com/course/viewer#!/c-ud727-nd/1-5455061279/m-638188663>

The advantages of Gaussian Mixture Models do not apply to this problem. Neither information about covariance is asked nor a confidence ellipsoid. Computation time is not critical more eighth. I go with K-Means as the “general-purpose” clustering algorithm.

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

```
In [6]: # TODO: First we reduce the data to two dimensions using PCA to capture variation
```

```
pca2 = PCA(n_components = 2)
reduced_data = pca2.fit(data).transform(data)
# http://scikit-learn.org/stable/auto\_examples/decomposition/plot\_pca\_vs\_lda.html#example-decom

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

```
#KMeans
clf_2 = KMeans(n_clusters = 2, init = 'k-means++', n_init = 10) #['k-means++', 'random' or an n
clf_3 = KMeans(n_clusters = 3, init = 'k-means++', n_init = 10)
clf_4 = KMeans(n_clusters = 4, init = 'k-means++', n_init = 10)
```

```

clf_5 = KMeans(n_clusters = 5, init = 'k-means++', n_init = 10)

#GMM
#clf_2 = GMM(n_components = 2, n_init = 10)
#clf_3 = GMM(n_components = 3, n_init = 10)
#clf_4 = GMM(n_components = 4, n_init = 10)
#clf_5 = GMM(n_components = 5, n_init = 10)

# http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans
# http://scikit-learn.org/stable/modules/generated/sklearn.mixture.GMM.html#sklearn.mixture.GMM

clusters_2 = clf_2.fit(reduced_data)
clusters_3 = clf_3.fit(reduced_data)
clusters_4 = clf_4.fit(reduced_data)
clusters_5 = clf_5.fit(reduced_data)

print clusters_2
print clusters_3
print clusters_4
print clusters_5

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)
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)
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)
KMeans(copy_x=True, init='k-means++', max_iter=300, n_clusters=5, n_init=10,
       n_jobs=1, precompute_distances='auto', random_state=None, tol=0.0001,
       verbose=0)

In [8]: # 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_2 = clusters_2.predict(np.c_[xx.ravel(), yy.ravel()])
Z_3 = clusters_3.predict(np.c_[xx.ravel(), yy.ravel()])
Z_4 = clusters_4.predict(np.c_[xx.ravel(), yy.ravel()])
Z_5 = clusters_5.predict(np.c_[xx.ravel(), yy.ravel()])

In [9]: # TODO: Find the centroids for KMeans or the cluster means for GMM

# for KMeans
centroids_2 = clf_2.cluster_centers_
centroids_3 = clf_3.cluster_centers_
centroids_4 = clf_4.cluster_centers_
centroids_5 = clf_5.cluster_centers_

```

```

# for GMM
#centroids_2 = clf_2.means_
#centroids_3 = clf_3.means_
#centroids_4 = clf_4.means_
#centroids_5 = clf_5.means_

print centroids_2
print centroids_3
print centroids_4
print centroids_5

[[ 4175.31101293 -211.15109304]
 [-24088.33276689 1218.17938291]]
[[-23978.86566553 -4445.56611772]
 [ 1341.31124554 25261.39189714]
 [ 4165.1217824 -3105.15811456]]
[[ 3542.08605212 -4936.7212132 ]
 [ 5710.98964991 12661.45687292]
 [-24220.71188261 -4364.45560022]
 [-14537.71774395 61715.67085248]]
[[ 5607.91709853 14199.18040025]
 [-9052.39957144 -4808.55909102]
 [-14537.71774395 61715.67085248]
 [ 6399.7117556 -4169.29690862]
 [-37704.64157991 -5488.35405895]]

In [10]: # Put the result into a color plot
Z = Z_2.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_2[:, 0], centroids_2[:, 1],
            marker='x', s=169, linewidths=3,
            color='w', zorder=10)
plt.title('Clustering on the wholesale grocery dataset (PCA-reduced data)\n'
          '2 Centroids are marked with white cross')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)

plt.xticks(())
plt.yticks(())
plt.show()

#---
Z = Z_3.reshape(xx.shape)
plt.figure(2)
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_3[:, 0], centroids_3[:, 1],
            marker='x', s=169, linewidths=3,
            color='w', zorder=10)

plt.title('Clustering on the wholesale grocery dataset (PCA-reduced data)\n'
          '3 Centroids are marked with white cross')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)

plt.xticks(())
plt.yticks(())
plt.show()

#---
Z = Z_4.reshape(xx.shape)
plt.figure(3)
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_4[:, 0], centroids_4[:, 1],
            marker='x', s=169, linewidths=3,
            color='w', zorder=10)

plt.title('Clustering on the wholesale grocery dataset (PCA-reduced data)\n'
          '4 Centroids are marked with white cross')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)

plt.xticks(())
plt.yticks(())
plt.show()

#---
Z = Z_5.reshape(xx.shape)
plt.figure(4)
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_5[:, 0], centroids_5[:, 1],
            marker='x', s=169, linewidths=3,
            color='w', zorder=10)

plt.title('Clustering on the wholesale grocery dataset (PCA-reduced data)\n'
          '5 Centroids are marked with white cross')
plt.xlim(x_min, x_max)

```



```
plt.ylim(y_min, y_max)
```

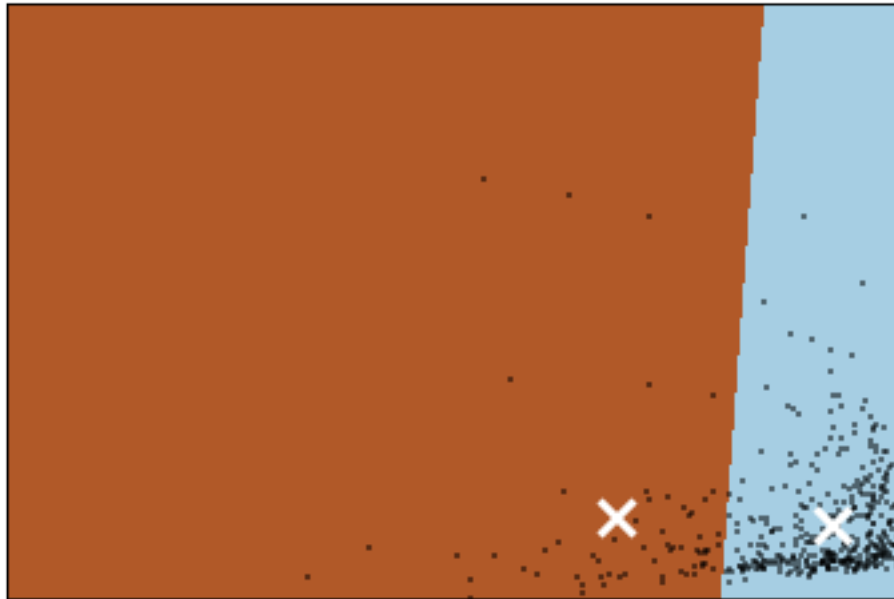
```
plt.xticks()
```

```
plt.yticks()
```

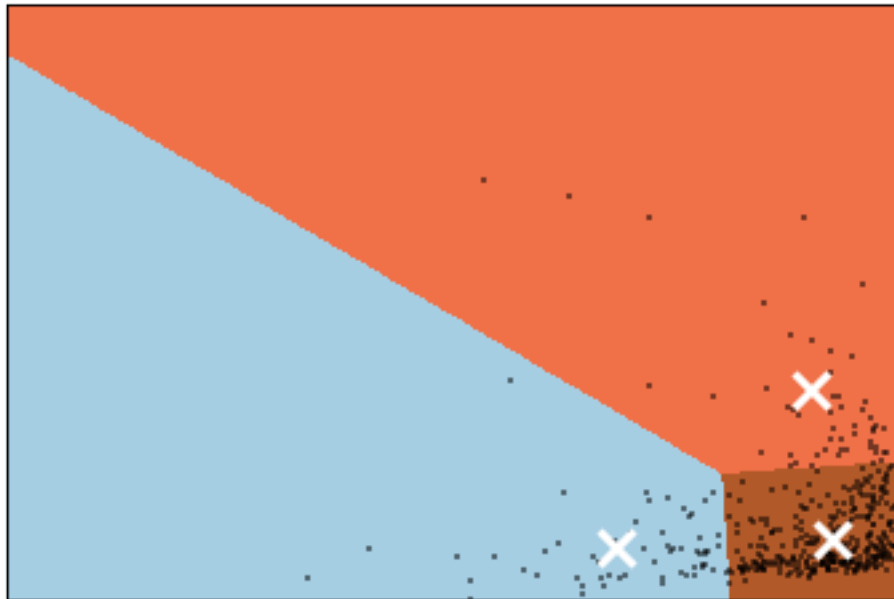
```
plt.show()
```

```
/Users/philipp/anaconda/lib/python2.7/site-packages/matplotlib/collections.py:590: FutureWarning: elementwise comparisons between arrays and scalars are deprecated  
if self._edgecolors == str('face'):
```

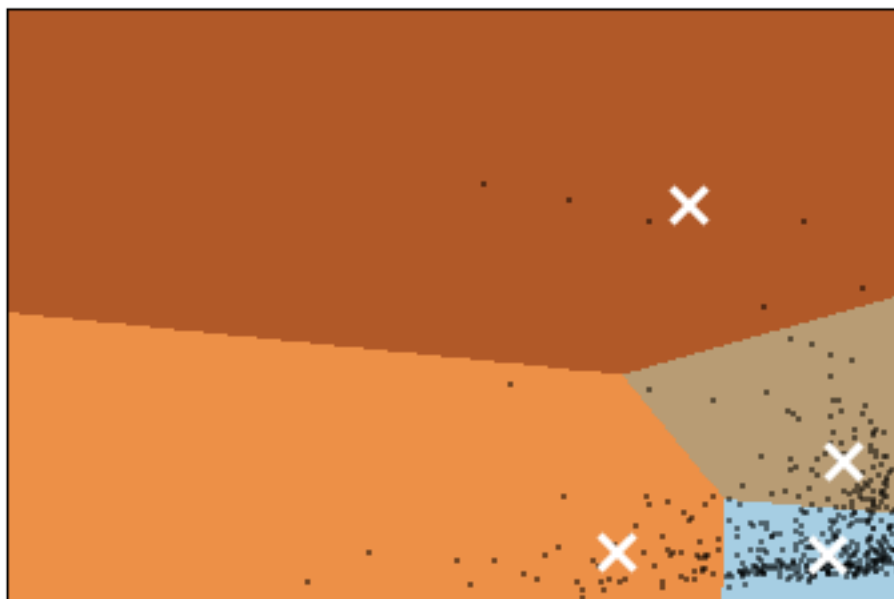
Clustering on the wholesale grocery dataset (PCA-reduced data)
2 Centroids are marked with white cross



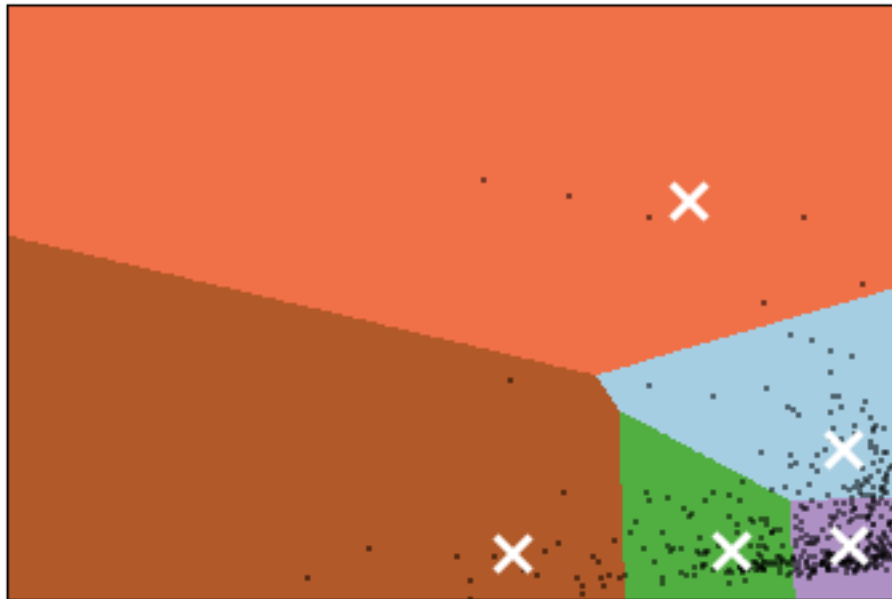
Clustering on the wholesale grocery dataset (PCA-reduced data)
3 Centroids are marked with white cross



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



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



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

Answer:

Having only two components to describe the clusters and considering the distribution of the customers in the plot, it makes not much sense to go beyond four clusters. The main difference between the four cluster plot and the three cluster plot is that the two (not very distinct) upper clusters are “merged”. For a two cluster analysis, one component would be sufficient.

Given that we saw that the first two components beeing relevant to the analysis. And the fact that a fourth cluster does not add significance to the plot. Three clusters seem to be the best choice for this data set.

lower-right cluster: Customers that order perishable and fast moving consumer goods. Like super-markets of different sizes.

lower-left cluster: Customers that mostly order perishable goods. Like fruit and vegetable outlets or central market.

upper cluster: Customers that mostly order fast moving consumer goods. Like small corner stores and kiosks.

In [11]: # Log transform

```
# from sklearn.preprocessing import FunctionTransformer
# FunctionTransformer throughs an ImportError: cannot import name FunctionTransformer

# Transformer
#transformer = sklearn.preprocessing.FunctionTransformer(np.log1p)
data_33 = data
#transformer.transform(data_33)

# PCA
pca3 = PCA(n_components = 2)
reduced_data_33 = pca3.fit(data_33).transform(data_33)
```

```

# Clusters
clf_33 = KMeans(n_clusters = 3, init = 'k-means++', n_init = 10)
clusters_33 = clf_33.fit(reduced_data)
print clusters_33

# Grid
x_min, x_max = reduced_data_33[:, 0].min() - 1, reduced_data_33[:, 0].max() + 1
y_min, y_max = reduced_data_33[:, 1].min() - 1, reduced_data_33[:, 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))

#
Z_33 = clusters_33.predict(np.c_[xx.ravel(), yy.ravel()])

#
centroids_33 = clf_33.cluster_centers_
print centroids_33

#

Z = Z_33.reshape(xx.shape)
plt.figure(2)
plt.clf()
plt.imshow(Z, interpolation='nearest',
            extent=(xx.min(), xx.max(), yy.min(), yy.max()),
            cmap=plt.cm.Paired,
            aspect='auto', origin='lower')

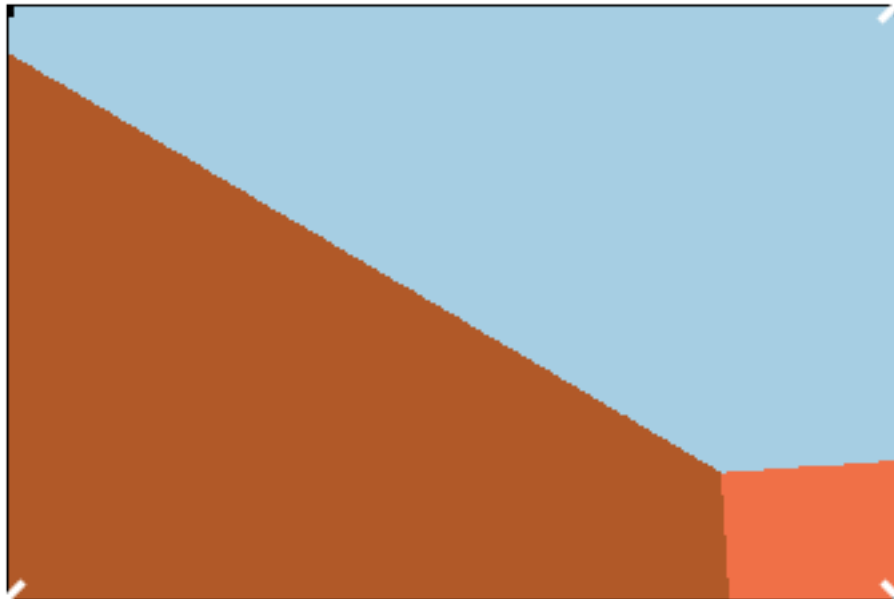
plt.plot(reduced_data[:, 0], reduced_data[:, 1], 'k.', markersize=2)
plt.scatter(centroids_3[:, 0], centroids_3[:, 1],
            marker='x', s=169, linewidths=3,
            color='w', zorder=10)
plt.title('Clustering on the wholesale grocery dataset (PCA-reduced data)\n'
          'using log-scale\n'
          '3 Centroids are marked with white cross')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xscale('log') #sub2 addition
plt.yscale('log') #sub2 addition
plt.xticks(())
plt.yticks(())
plt.show()

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)
[[ 1339.44615464 25546.49074629]
 [ 4114.95375632 -3081.03219608]
 [-24220.71188261 -4364.45560022]]

/Users/philipp/anaconda/lib/python2.7/site-packages/matplotlib/image.py:359: UserWarning: Images are not
warnings.warn("Images are not supported on non-linear axes.")

```

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



```
In [12]: #Using PC 2 and 3 for more insight
pca3 = PCA(n_components = 3)
reduced_data_22 = pca3.fit(data).transform(data)
reduced_data_22 = reduced_data_22[:,1:3]

In [13]: # The visualizer below assumes your clustering object is named 'clusters'

clf_22 = KMeans(n_clusters = 3, init = 'k-means++', n_init = 10) #{'k-means++', 'random' or an

# http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster
# http://scikit-learn.org/stable/modules/generated/sklearn.mixture.GMM.html#sklearn.mixture.GM

clusters_22 = clf_22.fit(reduced_data_22)

print clusters_22

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 [14]: # Plot the decision boundary by building a mesh grid to populate a graph.
x_min_2, x_max_2 = reduced_data_22[:, 0].min() - 1, reduced_data_22[:, 0].max() + 1
y_min_2, y_max_2 = reduced_data_22[:, 1].min() - 1, reduced_data_22[:, 1].max() + 1
hx_2 = (x_max_2-x_min_2)/1000.
hy_2 = (y_max_2-y_min_2)/1000.
xx_2, yy_2 = np.meshgrid(np.arange(x_min_2, x_max_2, hx_2), np.arange(y_min_2, y_max_2, hy_2))

# Obtain labels for each point in mesh. Use last trained model.
Z_22 = clusters_22.predict(np.c_[xx_2.ravel(), yy_2.ravel()])
```

```

In [15]: # for KMeans
centroids_22 = clf_22.cluster_centers_

In [16]: # Put the result into a color plot
Z_22 = Z_22.reshape(xx.shape)
plt.figure(1)
plt.clf()
plt.imshow(Z_22, interpolation='nearest',
           extent=(xx_2.min(), xx_2.max(), yy_2.min(), yy_2.max()),
           cmap=plt.cm.Paired,
           aspect='auto', origin='lower')

plt.plot(reduced_data_22[:, 0], reduced_data_22[:, 1], 'k.', markersize=2)
plt.scatter(centroids_22[:, 0], centroids_22[:, 1],
           marker='x', s=169, linewidths=3,
           color='w', zorder=10)
plt.title('Clustering on the wholesale grocery dataset (PCA-reduced data)\n'
         'using PC 2 and 3\n'
         'Centroids are marked with white cross')
plt.xlim(x_min_2, x_max_2)
plt.ylim(y_min_2, y_max_2)
plt.xticks(())
plt.yticks(())
plt.show()

```

Clustering on the wholesale grocery dataset (PCA-reduced data)
using PC 2 and 3
Centroids are marked with white cross



1.2.2 Conclusions

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

Answer:

On a very basic level, the 2D plotting of the PCA reduced samples gave me the most inside overall. I gained further inside by varying the number of centroids/clusters.

The plot of PC two and three looks very symmetric (except for some outliers). It does not add much information concerning the clustering. It is the same clusters dragged out into another dimension. No interesting splits to see. All centroids almost on one line. Besides, it is hard to find a description for the third dimension. It is not clear which information is contained in there.

Neither a logarithmic scale nor the third component can add more inside into the clustering.

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

Answer:

Some techniques to design new experiments after the PCA clustering are:

- *First of all, you should make a **validity check** against third party data, if the results of your PCA are reasonable.*
- *It might make sense to accumulate more data about the customers in each cluster by running a **survey**, **focus groups** or a **UER** (user experience research) to understand their needs beyond the PCs.*
- *Use this data to do **A/A testing** inside each cluster to gather information about the variability inside the clusters. This helps to determine a robust baseline for the A/B tests.*
- ***Intra-cluster A/B testing** (test and control group from the same cluster) for testing new cluster specific products and services. Eg. disposal services for customers from the perishable cluster or VMI (vendor managed inventory) for the FMCG (fast moving consumer goods) cluster customers.*
- ***Inter-cluster A/B testing** (test and control group from different clusters) for testing new general (unspecific) products and services. Eg. offering replenishment by urban freight distribution providers.*
- *Make sure that your test and control group for these experiments are not secured by checking the included customers for features like company size, order size, and location.*

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

Answer:

You can make better predictions if you put a new customer into the designated cluster first and make your predictions from there. Instead of predicting from the whole set of customers i.e. the average customer. This information can be used for cross- and upselling offers as well as bulk discounts, depending on the cluster. Sells people can use this data to see if a certain customer has weak sells in a certain category, compared with the other “reference customers” in the cluster. They might be able to identify opportunities for new offers to these customers.

Another interesting experiment would be to track the developments over time. For single customers and the whole set. Maybe there is a customer-life-cycle, where they move through different clusters over time. It might be possible to make an algorithm that could learn this pattern and make predictions, how a new customer might evolve.