UDACITY

Logo

Back to Machine Learning Engineer Nanodegree

Creating Customer Segments

REVIEW | HISTORY

Requires Changes

1 SPECIFICATION REQUIRES CHANGES

This is a very solid analysis here and impressed with your answers. You have an excellent grasp on these unsupervised learning techniques. You just need to fine tune one section and you will be good to go, but should be a simple fix and great for learning the material even better. Keep up the great work!!

Data Exploration

Three separate samples of the data are chosen and their establishment representations are proposed based on the statistical description of the dataset.

Great analysis here and good justification for your samples here by using the mean values in the dataset. Would suggest doing this for *all* samples.

Also note that using the median/percentiles would be much more appropriate than mean, since the median/percentiles are more robust to outliers, which we have here. But nice job!

Another really cool visualization you could make to analyze the distribution of purchasing behavior of your sample, would be with a Radar Plot

```
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
%matplotlib inline

scaler = MinMaxScaler()
df = np.round(samples, 1)
index  = df.index[:]
categories = list(df)
df = scaler.fit_transform(df)*100
N = len(categories)
angles = [n / float(N) * 2 * np.pi for n in range(N)]
angles += angles[:1]

plt.figure(figsize=(20, 5))
def Radar(index, title, color):
    ax = plt.subplot(1, 3, index+1, polar=True)
    ax.set_theta_offset(np.pi/2)
    ax.set_theta_direction(-1)
    plt.xticks(angles[:-1], categories, color ='grey', size = 8)
    plt.yticks((25, 50, 75, 100), ("1/4", "1/2", "3/4", "Max"), color = "
grey", size = 7)
    values = df[index]
    values = np.append(values, values[:1])
    ax.plot(angles, values, color = color)
```

```
    ax.fill(angles, values, color=color, alpha=0.5)
    plt.title('Sample {}'.format(title), y= 1.1)

for i, n in enumerate(index):
    Radar(index=i, title=n, color='r')
```

A prediction score for the removed feature is accurately reported. Justification is made for whether the removed feature is relevant.

> " It shows 'Fresh' provides unique information that cannot be perdicted by others; therefore, this feature is necessary when identifying customers' spending habits."

Exactly! Based on this model, Fresh is an independent feature, so necessary. Thus if we have a high r^2 score(high correlation with other features), this would not be good for identifying customers' spending habits(since the customer would purchase other products along with the one we are predicting, as we could actually derive this feature from the rest of the features). Therefore a negative / low r^2 value would represent the opposite as we could identify the customer's specific behavior just from the one feature.

Student identifies features that are correlated and compares these features to the predicted feature. Student further discusses the data distribution for those features.

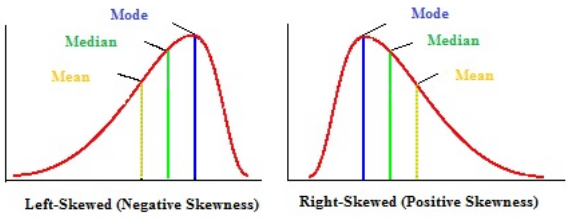
Great job capturing the correlation between features. Maybe also add correlation to the plot as well with

```
axes = pd.scatter_matrix(data, alpha = 0.3, figsize = (14,8), diagonal =
'kde')
corr = data.corr().as_matrix()
for i, j in zip(*np.triu_indices_from(axes, k=1)):
axes[i, j].annotate("%.3f" %corr[i,j], (0.8, 0.8), xycoords='axes fractio
n', ha='center', va='center')
```

And good ideas regarding the data distributions with your comment of

> "they both strongly right/positive skewed and most of data falls near 0. Besides, I listed all mean and medium values above and it is obvious that all mean values are larger than medium; which means there are some outlier points that try to pull the mean to the right part and cause the imbalance between mean and medium values."

Skewed right is correct. Could also mention log normal. And yes, we can actually get an idea of this from the basic stats of the dataset, since the mean is above the median for all features. We typically see this type of distribution when working with sales or income data.



Data Preprocessing

Feature scaling for both the data and the sample data has been properly implemented in code.

Student identifies extreme outliers and discusses whether the outliers should be removed. Justification is made for any data points removed.

Great job discovering the indices of the five data points which are outliers for more than one feature of `[65, 66, 75, 128, 154]`.

Outlier removal is a tender subject, as we definitely don't want to remove too many with this small dataset. But we definitely need to remove some, since outliers can greatly affect distributions, influence a distance based algorithm like clustering and/or PCA! You are correct that the loss function of the K-means algorithm is defined the terms of sum-of-squared distances, making it sensitive to outliers. In an attempt to reduce the loss function, the algorithm would move a centroid away from the true center of a cluster towards the outlier. This is clearly not the behavior we want.

One cool thing about unsupervised learning is that we could actually run our future analysis with these data points removed and with these data points included and see how the results change.

(http://www.theanalysisfactor.com/outliers-to-drop-or-not-to-drop/)
(http://graphpad.com/guides/prism/6/statistics/index.htm?stat_checklist_identifying_outliers.htm)

Maybe also examine these duplicate data points further with a heatmap in the original data.

```
# Heatmap using percentiles to display outlier data
import matplotlib.pyplot as plt
import seaborn as sns
percentiles = data.rank(pct=True)
percentiles = percentiles.iloc[outliers]
plt.title('Multiple Outliers Heatmap', fontsize=14)
heat = sns.heatmap(percentiles, annot=True)
display(heat)
```

Feature Transformation

The total variance explained for two and four dimensions of the data from PCA is accurately reported. The first four dimensions are interpreted as a representation of customer spending with justification.

Nice work with the cumulative explained variance for two and four dimensions.

- As with two dimension we can easily visualize the data(as we do later)
- And with four components we retain much more information(great for new features)

And good analysis of these PCA components. As always remember that the sign of the features in the component really wouldn't matter too much, since if we multiply the entire PCA dimension by -1 it would still be the same PCA component(so in PCA3 Fresh and Deli could be switched!).

To go even further here with the interpretation of the PCA components:

- In terms of customers, since PCA deals with the variance of the data and the correlation between features, the first component would represent that we have some customers who purchase a lot of Milk, Grocery and Detergents_Paper products while other customers purchase very few amounts of Milk, Grocery and Detergents_Paper, hence spread in the data.

Pro Tip: You can also visualize the percent of variance explained to get a very clear understanding of the drop off between dimension. Here is a some starter code, as np.cumsum acts like `+=` in python.

```
import matplotlib.pyplot as plt
x = np.arange(1, 7)
plt.plot(x, np.cumsum(pca.explained_variance_ratio_), '-o')
```

PCA has been properly implemented and applied to both the scaled data and scaled sample data for the two-dimensional case in code.

Clustering

The Gaussian Mixture Model and K-Means algorithms have been compared in detail. Student's choice of algorithm is justified based on the characteristics of the algorithm and data.

Good comparison and choice in GMM, as I would choose the same. As we can actually measure the level of uncertainty of our predictions!

As the main two differences in these two algorithms are the speed and structural information of each:

Speed:

- K-Mean much faster and much more scalable
- GMM slower since it has to incorporate information about the distributions of the data, thus it has to deal with the co-variance, mean, variance, and prior probabilities of the data, and also has to assign probabilities to belonging to each clusters.

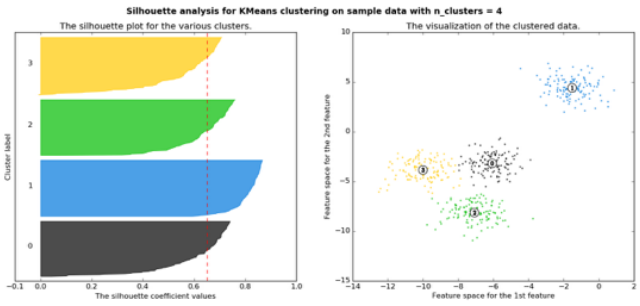
Structure:

- K-Means straight boundaries (hard clustering)
- GMM you get much more structural information, thus you can measure how wide each cluster is, since it works on probabilities (soft clustering)

Several silhouette scores are accurately reported, and the optimal number of clusters is chosen based on the best reported score. The cluster visualization provided produces the optimal number of clusters based on the clustering algorithm chosen.

Good work with the reversed for loop! As we can clearly see that K = 2 gives the highest silhouette score. Another cool interpretation method for Silhouette score is like this

(http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html)



The establishments represented by each customer segment are proposed based on the statistical description of the dataset. The inverse transformation and inverse scaling has been properly implemented and applied to the cluster centers in code.

Good justification for your cluster centroids by comparing the cluster centers with dataset percentile values. You could also examine the reduce PCA plot. Anything interesting about dimension 1 and how the clusters are split?

Pro Tip: We can also add the median values from the data and very easily visualize the cluster centroids with a pandas bar plot

```
true_centers = true_centers.append(data.describe().ix['50%'])
true_centers.plot(kind = 'bar', figsize = (16, 4))
```

Sample points are correctly identified by customer segment, and the predicted cluster for each sample point is discussed.

Excellent justification for your predictions by comparing the purchasing behavior of the sample to the purchasing behavior of the cluster centroid!

Could also check out the distance to each cluster centroid for justification

```
for i, pred in enumerate(sample_preds):
    print("Sample point", i, "predicted to be in Cluster", pred)
    print('The distance between sample point {} and center of cluster {}:
'.format(i, pred))
    print((samples.iloc[i] - true_centers.iloc[pred]))
```

Conclusion

Student correctly identifies how an A/B test can be performed on customers after a change in the wholesale distributor's service.

You have very good ideas here (this might actually be the *result* from A/B testing), but for this section can you also describe an A/B test and implement one for yourself. Think in terms of control / test groups and an experimental variable. Thus with this delivery change what could be the control group? What customers would be our test group? Thus can we use all of the segments together or do we need to run separate A/B tests on each independently?
https://en.wikipedia.org/wiki/A/B_testing#Segmentation_and_targeting
https://stats.stackexchange.com/questions/192752/clustering-and-a-b-testing

If you need a refresher of what an A/B test is and how to implement one, check out this link. It refers to a website A/B test, but the same terminology can be applied to any situation

- A/B Test

Student discusses with justification how the clustering data can be used in a supervised learner for new predictions.

Nice idea to use the cluster assignment as new labels. This refers to the idea of semi-supervised learning.

Another cool idea would be to use a subset of the newly engineered PCA components as new features(great for curing the curse of dimensionality). PCA is really cool and seem almost like magic at time. Just wait till you work with hundreds of features and you can reduce them down into just a handful. This technique becomes very handy especially with images. There is actually a handwritten digits dataset, using the "famous MNIST data" where you do just this and can get around a 98% classification accuracy after doing so. This is a kaggle competition and if you want to learn more check it out here KAGGLE

> "Another way I can come up with is that we compute the distance between the features of the new clients to the centers of each cluster and then pick up the closest cluster as the classification of the new client. Or in the GMM model, we only perform these E-steps for the new data points to include them in the clustering."

This is a great idea! We can also calculate something like the distance to the nearness to cluster center and using that as another feature!!

Rate this review

Comparison is made between customer segments and customer 'Channel' data. Discussion of customer segments being identified by 'Channel' data is provided, including whether this representation is consistent with previous results.

> "However, most data points though are somewhat ambiguous and do not clearly belong to one cluster or the other. There is considerable overlap in the true distribution."

Would agree! Real world data is really never perfectly linearly separable but it seems as our GMM algorithm did a decent job.

Maybe also fully examine how well the clustering algorithm did!

```
#find percentage of correctly classified customers
data = pd.read_csv("customers.csv")
data = data.drop(data.index[outliers]).reset_index(drop = True)
# might need to switch around the 0 and 1, based on your cluster seed
df = np.where(data['Channel'] == 2, 1, 0)
print("Percentage of correctly classified customers: {:.2%}".format(sum(d
f == preds)/float(len(preds))))
```

RESUBMIT PROJECT

DOWNLOAD PROJECT

Learn the best practices for revising and resubmitting your project.

RETURN TO PATH