UDACITY

Logout

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW | CODE REVIEW | NOTES

Requires Changes

3 SPECIFICATIONS REQUIRE CHANGES

SHARE YOUR ACCOMPLISHMENT

Data Exploration

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

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

> I attempted to predict Delicatessen spending based on the rest of the features. The reported prediction score was -0.429. This implies that this feature is necessary to predict customers' spending habits, because its value cannot be predicted from the rest of the features, therefore it contains non-redundant information.

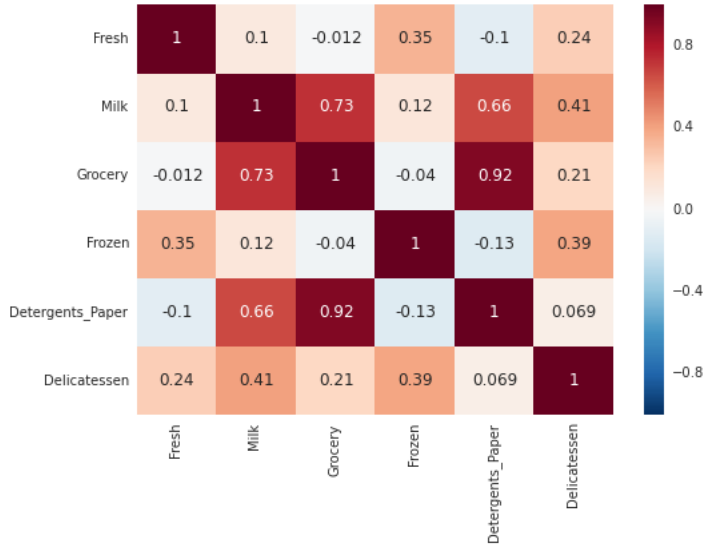
Correct, Delicatessen contains non-redundant information.

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

> A few pairs of features seems to exhibit some degree of correlation: (1) Grocery and Detergents, (2) Milk and Detergents, (3) Grocery and Milk.

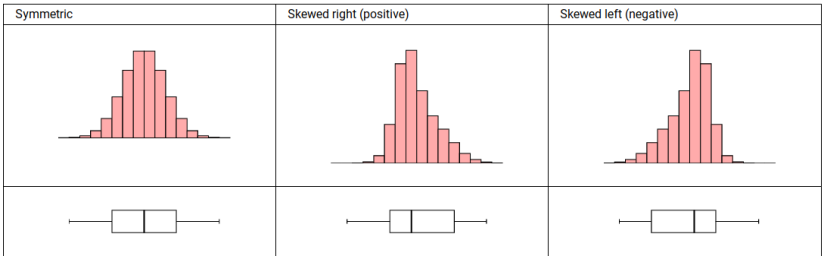
Correct, you can also visualize pairwise correlations as follows:

```
import seaborn as sns
sns.heatmap(data.corr(), annot=True)
```



> The data for all features seems to not be normally distributed. The curve is not symmetrical, and the mean seems skewed to the left, with a lot more points to the right of the mean than the left.

Correct, the distributions are positively skewed and they more closely resemble the log-normal distribution.



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.

> Yes, point 65, 66, 75, and 154 are outliers for multiple features.

Correct but there is yet one more such outlier, please make sure to report them all.

You may want to try using a counter:

```
from collections import Counter

c = Counter()

for feature in log_data.keys():

    Q1 = np.percentile(log_data[feature], 25)
    Q3 = np.percentile(log_data[feature], 75)
```

```
    step = (Q3 - Q1) * 1.5

    c.update(log_data[~((log_data[feature] >= Q1 - step) & (log_data[featur
e] <= Q3 + step))].index.values)

print [o for o in c.keys() if c[o]>1]
```

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.

> The first and second principal componens explain 71.42% of the variance in the data. The first four principal components explain 93% of the variance in the data.

Correct, the first four principal components capture most of the data.

> Third dimension: Represents high spending in delicatessen and low spending in fresh foods.

Correct, a principal component with dominant features that differ in their signs serves to identify customers that purchase a lot of the positively weighted categories but little of the negatively weighted categories as well as customers that buy a lot of latter but little of the former (such customers would score a large negative score in this dimension).

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

Clustering

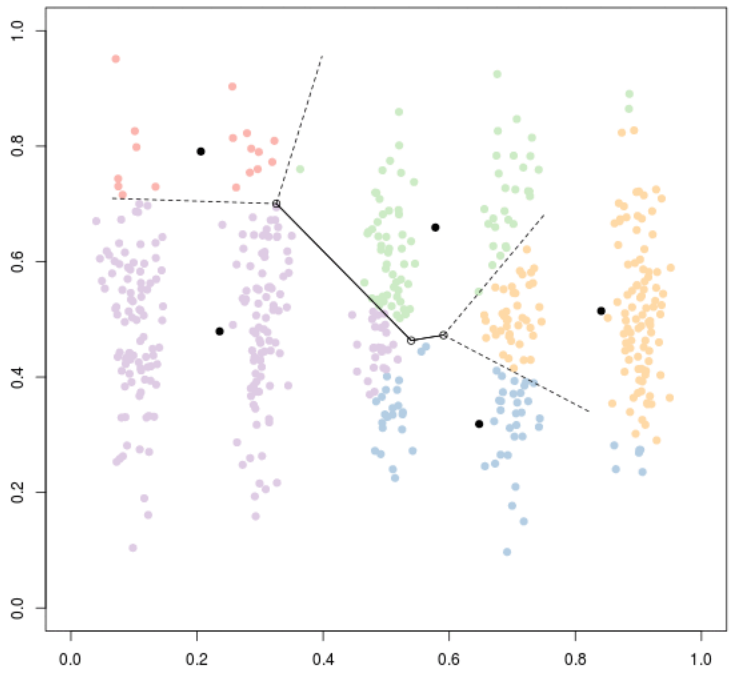
The Gaussian Mixture Model and K-Means algorithms have been compared in detail. Student's choice of algorithm is justified based on the characteristics of the algorithm and data.

> It works well for clusters that are convex and isotropic, but it responds poorly to elongated clusters, or manifolds with irregular shapes.

Correct, a big advantage of GMM over K-Means is the fact that it can model non-spherical clusters (all kinds of elliptical clusters to be precise) while K-Means assumes the clusters to be spherical.

The example from this blog post illustrates the point above.

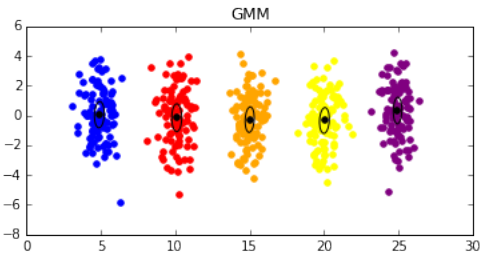
Below, K Means is applied to the following non-spherical clusters:



I've generated similar clusters and applied GMM (note that there is no restriction imposed on the structure of the co-variance to be used - a spherical structure would make GMM equivalent to K Means):

```
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
from matplotlib.patches import Ellipse
from sklearn import mixture


colors = [(1,'blue'),(2,'red'),(3,'orange'),(4,'yellow'),(5,'purple')]

clusters = []
for n,color in colors:
    cluster = multivariate_normal(
        mean = np.array([5*n,0]),
        cov = np.array([[0.4,0],[0,3.13]])).rvs(100)

    if n==1:
        X = cluster
    else:
        X = np.vstack((X,cluster))

    clusters.append(cluster)

g = mixture.GMM(n_components=5,covariance_type='full')
g.fit(X)

centroids = g.means_
covars  = g.covars_

#Plot an ellipses
for i in range(len(centroids)):
```

```
    mean = centroids[i]
    cov = covars[i]

    lambda_, v = np.linalg.eig(cov)
    lambda_ = np.sqrt(lambda_)

    ax = plt.subplot(111, aspect='equal')

    ell = Ellipse(xy=mean,
              width=lambda_[0], height=lambda_[1],
              angle=np.rad2deg(np.arccos(v[0, 0])))

    ell.set_facecolor('none')
    ax.add_artist(ell)

#Plot data
plt.title('GMM')
for i,color in colors:
    cluster = clusters[i-1]
    plt.scatter([c[0] for c in cluster],[c[1] for c in
cluster],color=color)


#Plot centroids
plt.scatter([c[0] for c in centroids],
    [c[1] for c in centroids],color='black')

plt.show()
```



Note how by modeling the co-variance of each cluster we can in turn model the shape of the cluster and avoid the complications that K Means runs into in this example.

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.

While using GMM you can also look at the Bayesian Information Criterion or the Akaike information criterion for finding the optimal number of clusters given that the model is fitted using maximum likelihood.

```
#Initialize two models
g2 = GMM(n_components=2)
g3 = GMM(n_components=3)

#Fit model to data
g2.fit(reduced_data)
g3.fit(reduced_data)

#BIC
print "2 clusters BIC: {bic}".format(bic=g2.bic(reduced_data))
print "3 clusters BIC: {bic}".format(bic=g3.bic(reduced_data))
```

```
#AIC
print "2 clusters AIC: {aic}".format(aic=g2.aic(reduced_data))
print "3 clusters AIC: {aic}".format(aic=g3.aic(reduced_data))
```

The rule of thumb with BIC and AIC is that the model with the lowest BIC or AIC is preferred. Note that BIC penalizes the number of free parameters (number of clusters in this case) more heavily than AIC. See the wikipedia articles for more details. Note as well that if while using either BIC and AIC we happen to come by two models (say 3 and 2 clusters) and the more complex model (3 clusters) happens to have a lower BIC/AIC value we might still prefer the less complex model (2 clusters) if the difference between their BIC/AIC value is small (about 2 points).

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.

> Segment 0 could represent traditional grocery stores which provide all types of goods but focus on groceries, dairy and detergents.

Remember to compare the mean cluster expenditures to the statistical description of the data set. For example, of which categories Segment 0's mean expenditures exceed the population's mean expenditure?

Try:

```
import seaborn as sns

sns.heatmap((true_centers-data.mean())/data.std(ddof=1), annot=True, cbar=F
alse, square=True)
```

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

> Samples points 0 and 1 would be best represented by Segment 0, which includes traditional grocery stores.

Don't forget to comment on whether each sample's expenditures actually mimic those of their assigned cluster (e.g. do they buy above the mean of the same categories of products?).

Conclusion

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

> The distributor may perform an A/B test by sending applying a 5 days/week delivery schedule to half the customers in Segment 0 and 3 days/week to the other half.

Correct, now that we have found two segments of customers we may perform two independent A/B tests within each of them.

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

> The distributor can use a supervised learner on the original data with the predicted customer segment engineered feature added to the data. The customer segment would be the target variable.

Correct, we can use the labels produced by the clusters as a new engineered feature to train our supervised learner.

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.

RESUBMIT PROJECT

DOWNLOAD PROJECT

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

Have a question about your review? Email us at review-support@udacity.com and include the link to this review.

RETURN TO PATH

Student FAQ