Feature Selection for Clustering

Review and testing of various techniques for feature selection before clustering and checking accuracy against Iris dataset

Gautam Rajeev

Problem Statement

1.1 Measuring performance of unsupervised learning:

The essence of machine learning algorithms lies in its ability to measure how well it is doing and then to improve on this measure. While measuring something like accuracy is straightforward in supervised learning, it is an interesting question when it comes to unsupervised learning. Here, it depends on what the goal of the algorithm is, for example, for clustering we want to be able to group the data points such that they form groups that are closely related and make intuitive sense. For this, we need to have a measure of how good the clustering is i.e are the data point within clusters close to each other (and away from other clusters). For k-means, we use the sum of squared distance (from cluster centroid) to converge on the best clusters and create an elbow chart from the sum of squared distance to estimate how many clusters is the most 'natural' solution.

1.2 Need for feature selection in clustering:

In all our lessons this semester for clustering using k-means or EM, we have generally used all the variables provided to us in the dataset. From our experience with supervised learning, we know that at least there sometimes is more beneficial to have a method to do variable selection get a model that is a better fit to new data. So, naturally the question comes up if the same is applicable for clustering i.e. will I get better clusters if I omit some of the features during clustering. Here, I first intend to show an example where feature selection is required for clustering using the Iris dataset and then test out various methods to carry out feature selection.

It is very important to note here that feature selection for unsupervised learning is very different and more challenging compared to devising a similar technique for supervised learning simply because we do not have the dependant variable and have no method to check if we have improved test accuracy once we carry out a selection process.

1.3 Dataset used:

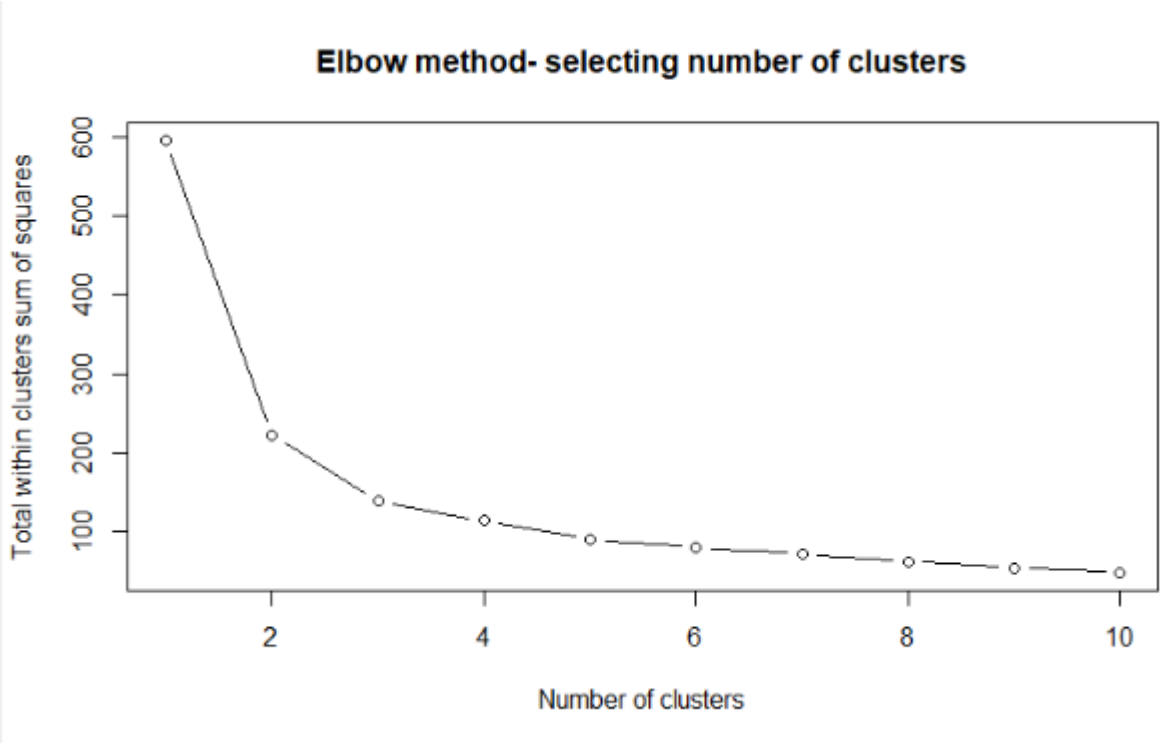
Throughout this work, I have used the dataset (https://archive.ics.uci.edu/ml/datasets/iris) which has 4 features and a 'class'- flower type. The feature are sepal width, sepal length, petal length and petal width. Ideally, on clustering the data points excluding the 'class' variables they should form the same clusters as the class. Here, we have a case where we already know what the ideal cluster results should be and we can cluster with various combinations of the 4 features to see if we get similar clusters with them:

Methodology

While clustering, we first need to decide how many clusters to create. This should be 3 as the dataset has 3 classes.

2.1 K- means and number of clusters

I ran k-means on the data set with all the variables after scaling all the variables and checked the within cluster sum of squares to the 'elbow curve' for the number of clusters.



We can see that this follows the expected pattern and we can see the elbow shape at 3 clusters.

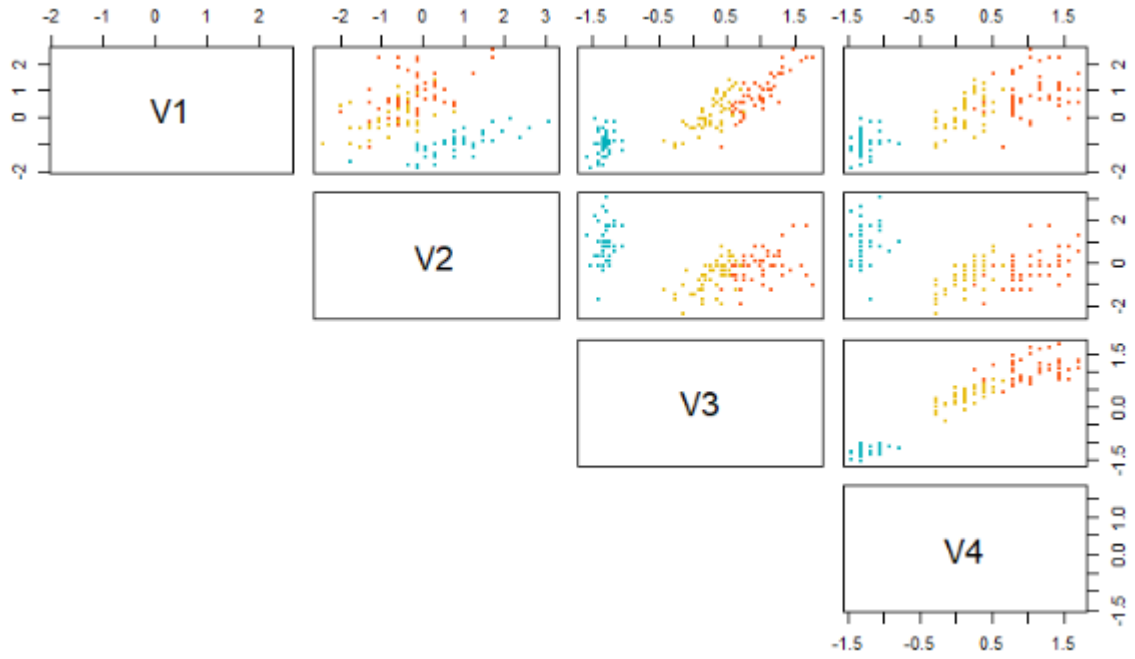
2.2 Creating 3 clusters with various feature combination

Now that we know that 3 is the optimum number of clusters from the elbow chart, our next step is check if using all features gives the best clustering result. In the dataset provided we have the feature classes given to us, so we can compare the results of our clustering against the feature classes provided to us and check the similarity to confirm if clusters are well defined. We can measure this by using the metrics generally used for measure classification scores like Recall, Precision and F measure.

First, I created all possible combinations of the feature that we have. We have 4 feature, so we have 2^4 i.e. 16 combinations. Excluding selecting none of the features; we have 15 possible combinations. I represented them with binary coding with 1 meaning that the variable has been selected to use for clustering and 0 means it has not been utilized by the clustering model. I ran k-means with all these possible combinations and compared the results with the actual class. I calculated metrics like Precision,Recall and F-Measure from this. I also calculated the 'within cluster sum of squared distance' and 'between cluster sum of squared distance'

V1	V2	V3	V4	Average_Precision	Average_Recall	F_Measure	within_ss	between_ss
0	0	1	1	0.960	0.960	0.960	17.907	280.093
0	0	0	1	0.960	0.960	0.960	8.456	140.544
0	0	1	0	0.948	0.947	0.948	7.867	141.133
1	0	1	1	0.872	0.867	0.870	62.621	384.379
0	1	1	1	0.863	0.860	0.862	94.683	352.317
1	0	0	1	0.836	0.833	0.834	50.801	247.199
1	1	1	1	0.834	0.833	0.834	138.888	457.112
0	1	0	1	0.822	0.820	0.821	78.707	219.293
1	1	0	1	0.810	0.807	0.808	124.497	322.503
1	1	1	0	0.809	0.807	0.808	118.345	328.655
1	0	1	0	0.805	0.807	0.806	42.748	255.252
1	1	0	0	0.777	0.773	0.775	101.931	196.069
0	1	1	0	0.772	0.767	0.769	74.617	223.383
1	0	0	0	0.721	0.707	0.714	22.981	126.019
0	1	0	0	0.633	0.573	0.602	27.686	121.314

The highlighted row is the case when all the features are selected. The first conclusion is obviously that it is not the best subset of features for this clustering exercise. It was much lower Precision, recall values than just selecting the last two variables- V3,V4 for clustering. In fact we just needed either of V4,V3 to get a much better cluster. This shows that the feature selection can definitely improve the clustering. We can take a look at the scatter plots of the features to get an idea of why this is happening.



On comparing the graphs of V4,V3 combination and V1,V2 combination; we can see clearly that there is clear separation of clusters for V3,V4 but this is not the case in V1,V2.

The other big insight is the values of the sum of squared distances for the various feature subset clusters. Within ss is the sum of squared distances between the points within the same cluster while the between ss is the separability- the distance between each cluster. K means works by minimizing the within ss value. However when we compare across feature subsets, the within ss value is dependant on the number of features selected. We can see that the subset with all the features has the highest within ss, value and highest between

ss value. The feature subsets with the less features has the lowest sum of squared distance value. This is sense as the distance measure here is Eucledian distance:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

As the number of features increases, the value inside the sqaure root increases which means that the sum of squared distances increases on adding features. This may cause it to be more than the distance value for a smaller feature subset even if it has worse clusters. In fact, Narendra and Fukunga [1] showed that the separability criterion monotonically increases with dimension (even when the new features are correlated or uncorrelated) as long as the clustering assignments remain the same. Hence, we need to find a method for normalizing for the number of features and that is what we will be trying to achieve through the rest of the report.

This is true even for the maximum likelihood criterion used in the EM algorithm, this has been shown by Jennifer G. Dy and Carla E. Brodley[2]

2.3 Sum of square ratio

One of the simplest ways to negate this is to use a ratio of the sum of squares to look at the best features. I have defined a new measure ss ratio which is equal to 'between sum of squares' divided by 'within sum of squares'. This is because we need to increase the between sum of sqaure value and decrease the within sum of squares value. This new measure is dimensionless and does not increase monotonically with the number of features.

V1	V2	V3	V4	Average_Precision	Average_Recall	F_Measure	within_ss	between_ss	ss_ratio
0	0	1	1	0.960	0.960	0.960	17.907	280.093	15.642
0	0	0	1	0.960	0.960	0.960	8.456	140.544	16.620
0	0	1	0	0.948	0.947	0.948	7.867	141.133	17.939
1	0	1	1	0.872	0.867	0.870	62.621	384.379	6.138
0	1	1	1	0.863	0.860	0.862	94.683	352.317	3.721
1	0	0	1	0.836	0.833	0.834	50.801	247.199	4.866
1	1	1	1	0.834	0.833	0.834	138.888	457.112	3.291
0	1	0	1	0.822	0.820	0.821	78.707	219.293	2.786
1	1	0	1	0.810	0.807	0.808	124.497	322.503	2.590
1	1	1	0	0.809	0.807	0.808	118.345	328.655	2.777
1	0	1	0	0.805	0.807	0.806	42.748	255.252	5.971
1	1	0	0	0.777	0.773	0.775	101.931	196.069	1.924
0	1	1	0	0.772	0.767	0.769	74.617	223.383	2.994
1	0	0	0	0.721	0.707	0.714	22.981	126.019	5.484
0	1	0	0	0.633	0.573	0.602	27.686	121.314	4.382

We can see that this new ratio is more useful than just looking at the within sum of squares or between sum of squares. It has a high value for the first 3 feature selections which have the best accuracy; however apart from the first 3, the ratio seems more affected by the variability in the within ss and the between ss values.

2.4 Cross projection of clusters

The next method we look at is based on the work of Jennifer G. Dy and Carla E. Brodley[2]. To compare between two feature subsets, we first create the clusters on the feature subsets and then we project the results of the clustering into each others subspaces to see if the sum of sqaures value decreases. While we can be sure that the 'within sum of square' value will be minimum for the original feature subspace (as the k-means algorithm minimizes on this), the value of each individual feature clusters projected and the decrease in

the values on removing features should give us more insight and we could use this as the basis for a greedy algorithm to select the features.

In the table below, I have projected all the clusters onto the subspace containing all the vectors and checked the cluster separation indices.

V1	V2	V3	V4	Average_Precision	Average_Recall	F_Measure	within_ss	between_ss	ss_ratio
1	1	1	1	0.834	0.833	0.834	128.597	138.888	1.080
1	1	0	1	0.810	0.807	0.808	129.343	273.497	2.115
1	1	1	0	0.809	0.807	0.808	129.781	267.345	2.060
0	1	1	1	0.863	0.860	0.862	130.298	243.683	1.870
1	0	1	1	0.872	0.867	0.870	130.858	211.621	1.617
1	1	0	0	0.777	0.773	0.775	131.105	399.931	3.050
0	1	0	1	0.822	0.820	0.821	132.812	376.707	2.836
0	0	1	1	0.960	0.960	0.960	133.985	315.907	2.358
0	1	1	0	0.772	0.767	0.769	134.781	372.617	2.765
1	0	0	1	0.836	0.833	0.834	134.851	348.801	2.587
0	0	1	0	0.948	0.947	0.948	135.683	454.867	3.352
1	0	1	0	0.805	0.807	0.806	135.789	340.748	2.509
0	0	0	1	0.960	0.960	0.960	135.842	455.456	3.353
1	0	0	0	0.721	0.707	0.714	156.452	469.981	3.004
0	1	0	0	0.633	0.573	0.602	202.191	474.686	2.348

As expected, the feature subset with the lowest within ss is the one with all the features as this is the subset we projected all clusters to. However, this is not the best subset in terms of accuracy On looking at the projections of the individual features, we see the expected results with V1 and V2 doing poorly but V3,V4 doing well.

However on comparing the projections with all features and on removing the important feature (V4,V3) - it does not give the expected result (removing V3,V4 seem to be better than removing V1,V2). This might be because of some relationship between the distribution of the variables that we haven't fully explored. This does not seem to be useful for backward elimination type method for feature selection.

2.5 Normalizing for the dimensions:

This method is taken from Jennifer G. Dy and Carla E who look at the entropy of the clusters and devise a method to normalise for the features. The essence is that the instead of looking at the Eucledian distance for the the within sum of squares, they normalise this by dividing with the maximum distance between points for each dimension first. The distance is given by the formula

$$D_{i_1,i_2} = \left[\sum_{k=1}^{M} \left(\frac{x_{i_1}k - x_i,k}{\max_k - \min_k}\right)^2\right]^{1/2}$$

where k is each dimension and M is the number of points.

V1	V2	V3	V4	Average_Precision	Average_Recall	F_Measure	within_ss
0	0	1	0	0.948	0.947	0.948	7.708
0	0	0	1	0.960	0.960	0.960	9.057
0	0	1	1	0.960	0.960	0.960	9.435
1	0	1	0	0.805	0.807	0.806	12.316
1	0	1	1	0.872	0.867	0.870	13.160
0	1	1	0	0.772	0.767	0.769	13.848
0	1	1	1	0.863	0.860	0.862	13.922
0	1	0	0	0.633	0.573	0.602	14.112
1	0	0	0	0.721	0.707	0.714	14.274
1	1	1	0	0.809	0.807	0.808	14.442
1	0	0	1	0.836	0.833	0.834	14.578
1	1	1	1	0.834	0.833	0.834	14.884
1	1	0	0	0.777	0.773	0.775	14.953
0	1	0	1	0.822	0.820	0.821	15.078
1	1	0	1	0.810	0.807	0.808	16.058

We see that while it manages to accurately capture the importance of V3,V4 compares to the other features; it is slightly skewed towards the features subsets with lesser number of features. The subspaces with just V3, V4 show lower within ss than the others. However, it is an improvement on the other two methods. It is dependant on the range of the variable distribution (though we normalized the variables in our case) and on each variables individual distribution within the range i.e. if some variables have a sharp multi-modal distributions such that the distance between these points divided by the maximum range is very low, then it can affect the distance metric adversely.

2.5 Using PCA to normalize:

The last method is to use PCA to normalize the clustering technique. In this , I first use PCA to reduce all the feature subspaces into their first eigen value projection and then cluster on this compare the results.

V1	V2	V3	V4	Average_Precision	Average_Recall	F_Measure	within_ss	between_ss	ss_ratio
0	1	1	1	0.9466667	0.9466667	0.9466667	12.343281	318.6577	25.816290
0	0	1	1	0.9600000	0.9600000	0.9600000	13.170984	279.2960	21.205399
0	0	1	0	0.9484702	0.9466667	0.9475676	7.867216	141.1328	17.939355
1	1	1	1	0.9267707	0.9266667	0.9267187	23.200232	411.6559	17.743613
0	0	0	1	0.9604701	0.9600000	0.9602350	8.456319	140.5437	16.619960
1	1	0	1	0.8867547	0.8866667	0.8867107	16.612350	273.0707	16.437812
1	1	1	0	0.8411165	0.8400000	0.8405579	19.405790	281.7873	14.520783
1	0	1	1	0.8729277	0.8666667	0.8697859	32.310284	380.3812	11.772759
1	0	0	1	0.8498098	0.8466667	0.8482353	26.398821	244.4744	9.260808
0	1	1	0	0.7317729	0.7266667	0.7292109	22.021369	190.8162	8.665047
0	1	0	1	0.7821821	0.7800000	0.7810895	21.164106	182.3887	8.617830
1	0	1	0	0.8390063	0.8333333	0.8361602	29.136496	249.7548	8.571889
1	1	0	0	0.7402402	0.7200000	0.7299798	18.717219	147.8007	7.896508
1	0	0	0	0.7212658	0.7066667	0.7138916	22.981287	126.0187	5.483536
0	1	0	0	0.6330672	0.5733333	0.6017214	27.685577	121.3144	4.381864

We can see that on sorting by the ss ration, we manage to almost match the accuracy statistics except for the combination V3,V4 which is the best and (V3,V1), (V3,V2). Regardless, considering that we are comparing using an unsupervised results, these are much better than where we started from and are much better than the other techniques explored here.

Conclusions:

We tried out various methods to better estimate what could be a good feature subset for clustering by comparing against actual classification and made a lot of progress compared to looking at just within ss. Normalizing for the dimensions with the use of PCA proved to show the best results, however we must keep in mind that we did this only on the Iris dataset and we must try this on other distributions. There is no doubt that using these methods certainly give some insight into the best feature and are certainly better than just looking at the within ss, between ss values. However, more exploration is required to check performance for other distributions and datasets.

References:

1. A branch and bound algorithm for feature subset selection - Narendra and Fukunga
2. Feature Selection for Unsupervised Learning- Jennifer G. Dy and Carle E Brodley
3. Feature Selection for Clustering - Manoranjan Dash and Han Lui