K-Means Clustering EViews add-in

Erhard Menker

June 6, 2017

Document Overview

This document explains the theory & motivation behind k-means clustering, and presents the application of the algorithm using its EViews add-in. The core of the algorithm's implementation adheres to that given by Dr. Andrew Ng's Stanford machine learning course.[1] However, any implementation deviations from the pseudocode are of course the responsibility of the add-in's author. Inquiries of any kind are warmly welcomed; the add-in's author, Erhard Menker, can be reached via email (ejmenker@gmail.com), and the working version of the project will continue to be developed on Github.[2]

Cluster Analysis Motivation

Cluster analysis is an unsupervised machine learning algorithm, meaning it is applied to "unlabeled data" (the classification of observations is not given in the dataset). Given a dataset of m observations, k-means clustering assigns each observation in the dataset to belong to one of cluster centroid 1 or 2 or … or k, conditioned that the # of centroids is less than m. A cluster's centroid is the coordinates of the arithmetic average of each series for its associated observations. This assignment is done by finding the centroids that minimizes the cost function, a measure that values centroids that have close proximity to their associated points. Therefore, clustering algorithms are a useful form of exploratory data analysis and can be used in EViews with time series data (e.g. classifying macroeconomic regimes over a country's history) & cross section data (e.g. customer segmentation).

User Arguments

- *Mandatory*
  - **k** – the # of centroids that the dataset with m observations will be partitioned into (constrained to be less than m)
- *Optional*
  - **quiet** – shut off the add-in's log messages
    - defaults to presenting log messages
  - **inits** – the # of solves of random initializations of the cluster centroids to occur
    - defaults to 3
  - **max_iters** – the maximum # of times the cluster centroids are allowed to move based on its associated observations for a given solve of centroids
    - defaults to 10
    - if set to "NONE", will continue until convergence occurs

[1] https://www.coursera.org/learn/machine-learning/lecture/93VPG/k-means-algorithm
[2] https://github.com/ErhardMenker/kMeans4EViews

- **series** – a space delimited string of the series on the workfile page to be included in the cluster analysis
  - defaults to including all series on the workfile page
- **smpl** – observations to be included in the clustering algorithm
  - defaults to the sample at time of function call
  - if @all | all are standalone arguments or set equal to smpl, then set sample to equal the range
- **impute** | **interpolate** – standalone argument that interpolates a series' missing values
  - defaults to not interpolating
  - for time series pages, linearly interpolate series
  - for unstructured workfiles, impute with the series' median

K-Means Algorithm Summary[3]

```
' preprocess series
' for each random initialization:
'       initialize k random observations as cluster centroids
'       while the cluster centroids continue to converge OR the max # of cluster moves is NOT reached:
'             find each observation's closest centroid
'             set each cluster centroid equal to the mean of its associated observations for all series
'       find each observation's closest centroid
'       calculate the cost function of the cluster solve
'       if the cost function is the smallest yet for all solutions:
'             store the centroids & their associated observation as the optimal clustering thus far
```

Add-in Output

- 2 objects are returned to the page on which the add-in is called:
  - **obs_cluster** – a series object where each observation states the cluster # to which the observation belongs
  - **kmeans_results** – a text object declaring, for each centroid, how the mean of each of its series compares to the overall series mean (both nominal & percentage differences)

[3] For a detailed description with formulas & visuals, see: http://cs229.stanford.edu/notes/cs229-notes7a.pdf

Example Application

Program:

```
' create an annual workfile
wfcreate(wf=k_means_example, page=DATA_A) a 1975 2016

' fetch series (accessed via FRED API)
' a) unemployment rate
fetch fred::unrate
' b) gdp yoy
fetch fred::a191ro1q156nbea
	rename a191ro1q156nbea gdp_yoy
' c) cpi yoy
fetch fred::cpiaucns
	series cpi_yoy = @pcy(cpiaucns)
	delete cpiaucns

' calculate 2 cluster centroids between 1975 & 2016
exec .\..\kmeans.prg(k = 2)
```

Output:

a. series object (*obs_cluster*) indicating which observations correspond to the 2 centroids

1975		**1982**	2	**1989**	1	**1996**	1	**2003**	1	**2010**	2
1976	1	**1983**	2	**1990**	1	**1997**	1	**2004**	1	**2011**	2
1977	1	**1984**	1	**1991**	2	**1998**	1	**2005**	1	**2012**	2
1978	1	**1985**	1	**1992**	1	**1999**	1	**2006**	1	**2013**	2
1979	1	**1986**	1	**1993**	1	**2000**	1	**2007**	1	**2014**	1
1980	2	**1987**	1	**1994**	1	**2001**	1	**2008**	2	**2015**	1
1981	2	**1988**	1	**1995**	1	**2002**	1	**2009**	2	**2016**	1

b. text file (*kmeans_results01*) stating how mean of each series for each centroid compares to the series' aggregate mean

```
*****************************************************************
*** ANALYSIS OF K-MEANS CLUSTERING RESULTS ***
*****************************************************************

*******************************************************************************
 *** USER SELECTED PARAMETERS ***

# of clusters: 2
# of iterations used: 3
Series included in clusters: CPI_YOY, GDP_YOY, UNRATE
*******************************************************************************

*******************************************************************************
CLUSTER 1:

CPI_YOY cluster mean is 3.482
   i) 0.268 units lesser than overall CPI_YOY mean
   ii) 7.14% lesser than overall CPI_YOY mean

GDP_YOY cluster mean is 3.511
   i) 0.697 units greater than overall GDP_YOY mean
   ii) 24.789% greater than overall GDP_YOY mean

UNRATE cluster mean is 5.756
   i) 0.651 units lesser than overall UNRATE mean
   ii) 10.165% lesser than overall UNRATE mean

*******************************************************************************
CLUSTER 2:

CPI_YOY cluster mean is 4.48
   i) 0.73 units greater than overall CPI_YOY mean
   ii) 19.473% greater than overall CPI_YOY mean

GDP_YOY cluster mean is 0.911
   i) 1.902 units lesser than overall GDP_YOY mean
   ii) 67.606% lesser than overall GDP_YOY mean

UNRATE cluster mean is 8.183
   i) 1.776 units greater than overall UNRATE mean
   ii) 27.723% greater than overall UNRATE mean

*******************************************************************************
```

Analysis:

This k-means application looks at annual U.S. macroeconomic data between 1975 & 2016. Level unemployment & year-over-year percentage changes in real GDP & CPI capture the state of the US macroeconomy over the past few decades. Choosing 2 clusters centroids means that there are going to be 2 groupings of the economy captured. Looking at the text file, we can see there is cluster 1 (higher GDP growth and lower unemployment level & CPI growth) and cluster 2 (lower GDP growth and higher unemployment level & CPI growth), with these comparisons to the overall mean derived from the text output.

These economic regimes correspond mostly to understood macroeconomic U.S. history; observations from cluster 2 appear during periods of U.S. recession including in the early 1980s & 1990s, and the financial crisis.

This serves as a vindication of accuracy. However, more interesting applications require increased sophistication of problem setup. For example, the economist could consider adding more macroeconomic series & higher frequency and increasing k to detect for more complicated regimes (e.g. high GDP growth, lower unemployment & higher inflation). Recessions are defined by GDP, so another application would be to remove GDP & see if different economic indicators, like manufacturing/trade data, could be used to pick up recessions (or even serve as leading indicators of recessions when the series are considered together).

To do analysis on 1 centroid of the data, conditional sample definitions can be called:

```
' constrain the sample to the recession values
smpl @all if obs_cluster = 2
```

Frequently Asked Questions

1) ***Isn't there an analytical algorithm to deterministically get the optimal clustering!?***

There is! It's called hierarchical clustering.[4] Hierarchical clustering requires only 1 solve since the returned solve is analytical & thus guaranteed to minimize the cost function. The reason it is not implemented is because of algorithmic inefficiency as the inputted series become arbitrarily large. Hierarchical clustering requires, for each unmerged point & centroid of merged points, merging into clusters based on the minimal pairwise distance across all these points & merged cluster centroids. A superficial hierarchical clustering implementation would run in quadratic time, and while divide-and-conquer techniques can speed up execution, k-means clustering is implemented because it runs in linear time. Executing multiple solves of a linear algorithm is preferred to executing one solve of a quadratic algorithm for a sufficiently large dataset, and only becomes more preferable as that dataset continues to grow.

2) ***k is the only mandatory argument…how should I choose it?***

As k increases (from 1 to the # of observations), the cost function monotonically decreases (if k were to equal the # of observations, the cost would be 0 as each observation would be explained perfectly with its very own centroid equal to itself). This means that the selection of k, if algorithmic, would have to implement an accurate measure that assesses when increasing k by 1 reduces the cost function by an insufficient amount. In addition to the challenge of discerning this inflection point, which could vary based on each problem's context, the algorithm could be computationally taxing because the k-means cluster would have to be solved for each candidate value of k.

The implemented approach is to not algorithmically choose k, but rather judge the usefulness of k based on the ability to serve purposes downstream with which the classification is used. For example, if an economist is using the add-in to classify macroeconomic regimes, she may find that a small value of k misses common regimes in the macroeconomy, while too large a value of k creates different clusters that substantively reflect a similar macroeconomy or just captures noise, and are thus based in inconsequential variations of series values.

3) ***How are the series preprocessed?***

a. Eliminate series that have all NAs or no variability over k-means sample
b. Normalize each series
c. Interpolate (if this argument is passed in to overwrite the default of no interpolation)
d. Disregard observations that have at least 1 series with an NA
e. Add a small # to each series value to avoid the problem of duplicate series creating pointless centroids

4) ***What does it mean for the series to be normalized? Does this distort results?***

Series are normalized by converting their values into their z-scores (the normalized value is equal to the original series' value minus the series' mean, with this difference divided by the series' standard deviation). This is essential; normalization removes the distortionary effects of comparing series with different magnitudes & measures of dispersion. In the absence of

[4] https://www.coursera.org/learn/exploratory-data-analysis/lecture/68OwU/hierarchical-clustering-part-1

normalization, relatively slight differences in one series could dwarf relatively major differences in another series if the magnitude of the former series is substantially larger than the latter. Normalization does not distort results – the normalized series are only used to determine how the observations should be clustered. Absolute & percentage changes presented in the outputted text file are all calculated based on the original series' values.

5) ***Can a cluster centroid wind up with no associated observations?***

Protections are put in place to protect against this in the implementation. Centroids are initialized to equal random observations, so each centroid is identical to a point that it will be associated with initially. An observation-less centroid could occur if multiple points are identical, but protections are taken against this: a very tiny random uniform is added to each normalized series' values, a linear in time approach to protect against identical points. Giving the small add factor all but guarantees that initial assignment is maintained for at least the 1^{st} iteration given duplicates. The interpretation of the outputted cluster only becomes problematic for contrived examples that probably shouldn't be clustered anyways (e.g. almost all the initial points are identical, so the clustering is solely contingent based on the minor add factor when there is in fact no real difference between most the data points).

6) ***Why eliminate series with no variability?***

If a series has no variability (all observations are the same), it adds no explanatory power to the dataset's observations. Considering the n other series, adding the series to the cluster is like mapping into n + 1 space where the $(n + 1)^{st}$ dimension value is just a constant real number. This added dimension adds nothing in differentiating the dataset's observations from one another because it's a constant attribute across points. Also, constant series cannot be normalized since normalized series are divided by the series' standard deviation, which is 0 in this case.

7) ***How does imputation impact results & when should it be used?***

Imputation is useful when a lot of information would be lost if not implemented. If there are many series and just one otherwise dense series has an NA for a given observation, imputation will prevent the observation from being discarded and all the other series' information lost. When results are presented, the imputed value will not be considered in the text file, as it is just used to determine the clustering of the observations among the normalized series, while means are calculated based on the original values at time of the add-in's call. However, if a series has many NAs, it may make sense to outright exclude the series from clustering or change the sample since the imputed values become more questionable.