SOCIETY OF ACTUARIES
PREDICTIVE ANALYTICS AND FUTURISM SECTION

PREDICTIVE ANALYTICS AND FUTURISM
APRIL 2020

Like Peas in a Pod: Ideas in Cluster Analysis

By Michael Niemerg

Cluster analysis is easy to grasp conceptually. You simply "group like things together." The fundamental algorithms like k-means and hierarchical clustering are also relatively easy to understand and don't require much background besides a little understanding of algebra. Despite such an intuitive premise, things can quickly become sophisticated. To that end, let's explore some extensions of cluster analysis beyond the vanilla approaches to understand ways we can improve our analysis and to get a broader sense of what's possible.

First, let's define more clearly what cluster analysis is. Cluster analysis is an unsupervised data-mining technique used to assign data into groupings whereby observations within a grouping (i.e., cluster) will be similar to each other. It is unsupervised because the algorithm does not have access to any label or response information when training. Instead, the model learns the relationships from the data itself without having access to any sort of "correct" answer (usually—we'll come back to this point).

Without a correct answer to judge the results by, how do we determine what makes a good cluster in the first place? This turns out to be a surprisingly tricky question. We are trying to create groupings that exhibit similar characteristics. Two natural consequences of this are that observations within a cluster should be similar to each other and observations within different clusters should be less similar to each other. While there are a variety of evaluation metrics in cluster analysis, most are ways of measuring and comparing these two ideas. The silhouette score is an illustrative example. For each data point, it measures the distance between the current data point and all the other observations within the same cluster and compares that to the average distance between the current data point and all observations for the nearest neighboring cluster. The closer the current data point is to the other observations within its own cluster center, and the farther away it is from the nearest cluster's observations, the better.

While in cluster analysis you don't use labels in training your model (with exceptions as noted later), you can be in the lucky situation of having access to a cluster assignment label to evaluate the model. If this is the case, there are a few methods that can evaluate cluster quality, many of which are closely related to the types of error metrics you would look at for classification tasks. Even if you don't have access to true cluster labels, another way to use this idea, called supervised clustering, is by using proxy features that you think the cluster model should be able to do a good job of classifying. For instance, you might want to group people with similar health profiles together. If you know what their health care claim costs are, you might be able to use this as a proxy of health.

Now that we've established some concepts behind cluster evaluation, we'll discuss some ideas that might be useful in practice. The ideas that follow are presented roughly in order of practicality.

DATA PREPROCESSING

Modeling is always about the data and its representation. Data is the foundation upon which everything rests. One of the first things to try, even before creating a model, is to make sure you preprocess your data in a meaningful way. Because clustering relies on the distance between features, you will often want to normalize all the data so that it is all represented on a similar scale. One way to do that is to standardize the features to be

within the range [-1,1]. This should effectively give every feature in your data set an equal "vote" in the model-building process. Without normalization, features with large absolute values will be overweighted. You might also do the opposite and transform the scales of certain columns to have more influence in the clustering algorithm. In adjusting the weights, you can express a preference for how large of a vote each feature gets in the model.

Another step that is beneficial is to perform label encoding for categorical features, i.e., represent your categories as numbers. To do this, you need to be a little thoughtful and treat ordinal and categorical data differently. If you have three distinct categories and you assign them to values 1, 2, and 3, then the clustering algorithm will treat two records coded with 1 and 2 as more similar than two records coded with 1 and 3. This might make sense if your data is naturally ordinal. If not, you might try one-hot encoding, where you create a single indicator feature for each category in the original feature.

Lastly, you can perform impact encoding. The idea here is that you replace each categorical value in a feature with a numerical value derived from some outcome of interest. For instance, if you have ZIP code as a feature it will have high cardinality and may not be useful in a model—it takes on too many values, mostly non-credible. You could instead create a feature that encodes the average income, mortality rate, or per member per month (PMPM) health care claim cost within that ZIP code.

One possible way to improve clustering model results is simply by trying out different model types.

ALGORITHMIC POTPOURRI

One possible way to improve clustering model results is simply by trying out different model types. A myriad of different algorithms exist out there for clustering—literally hundreds. Yet, just like in the supervised learning case, the number of really useful algorithms is much smaller. K-means and hierarchical clustering are the obvious choices. Gaussian mixture models, density-based spatial clustering of applications with noise (DBSCAN), spectral clustering, and clustering large applications (CLARA) are some more common alternatives. To select the right algorithm there are two options—learn about the differences between them and carefully select the appropriate algorithm based upon the problem at hand (for instance, k-medoids is more robust to outliers than k-means while spectral clustering is more resistant to noise), or simply try out multiple algorithms and evaluate the results. Given the lack of a ground-truth label, knowing when one technique is performing better than another requires considerable thought. It is not as straightforward as the supervised learning case.

FEATURE AND CLUSTER SELECTION

Irrelevant features in cluster analysis can slow down model training and unnecessarily bloat model size. Worse, they have the potential to degrade model performance. Feature selection can be tricky though and is a bit of a catch-22. If you already know what features matter, you would have already applied that knowledge and wouldn't need to do feature selection. And if you don't, and you lack a ground-truth label, how do you judge whether a feature matters?

Of course, the simplest form of feature selection is simply applying domain understanding. While qualitative, this form of feature selection is highly valuable. We can pick features that likely correlate to behaviors we are interested in and remove redundant, uninteresting, or highly correlated features. Barring that, there are also specialized algorithms that deal in feature selection. For instance, a feature selection algorithm might pick the features that most contribute to cluster compactness.

Intelligently selecting the number of clusters (for algorithms that don't do this automatically) is another way to improve the analysis. Several different metrics are useful, including the gap statistic, average silhouette score, and the elbow method. Conversely, the number of clusters selected may not involve any of these methods and may be informed by the use case. As an example: fifty clusters might be too unwieldy from an implementation perspective but five clusters might not be granular enough to give interesting insights.

CONSENSUS CLUSTERING

Consensus clustering is a way to (potentially) improve cluster analysis by creating multiple cluster models and then combining them. Effectively, consensus clustering is a way to create ensembles, similar to how we might for supervised learning (not surprisingly, this approach is also called ensemble clustering). The reason it works is similar to the intuition for ensembles in the supervised case, with a few new wrinkles added in. One of those wrinkles is that different clustering methods divide the feature space in very different ways geometrically, so combining algorithms can be a bit risky and isn't as easy as it might be in the supervised learning case.

There are several ways to create consensus models. Some possible methods include relabeling/voting, a co-occurrence matrix, and median partition methods. My experience is primarily with relabeling/voting methods so I'll expound upon that method for illustrative purposes.

One thing to remember with consensus clustering is that, unlike in supervised learning, there is no response value in clustering,

so the output of a cluster model is an arbitrary label, usually an integer. That is, there is no guarantee in any set of clustering models that the fifth cluster in one model corresponds to the fifth in another model. The label is just a placeholder, so you can't simply look up the fifth cluster in two different models and expect them to have any correspondence with each other.

Relabeling/voting takes care of this problem for us. First, it determines label correspondence between the base models. It does this by checking the labels and switching them so that the labels in all the base models of the ensemble refer to the same cluster. With all our labels on a consistent basis, we can now create the ensemble's prediction with either a hard or soft (probabilistic) classification of cluster membership for each observation. To do this we give each base cluster in the ensemble a vote as to which cluster it thinks the data point should belong to. For instance, if we had five different models with cluster predictions of 2, 2, 2, 3, and 2, then by way of voting we would label this instance as belonging to the second cluster for a hard classification. For a soft classification, we would view this point as having a 0.8 weight or probability of being in the second cluster and a corresponding weight of 0.2 of it being in the third cluster.

PREPROCESSING WITH SHAP VALUES

Let's now take things back a few steps, before applying any algorithms, to when we preprocessed our data. Instead of using the preprocessed data "as is," let's explore a new possibility: Shapley additive explanations (SHAP) values. SHAP values are derived from a classification or regression model. To explain how we can use SHAP values in clustering, let's first dive into what they are.

SHAP values were initially introduced as a model interpretation method, where we allocate how much each feature in a supervised learning task contributes to the difference between the prediction for that particular observation and the overall average. As an example, the SHAP value in a linear regression is actually the coefficient for that feature multiplied by the feature value—because this value in a regression equation tells us how to move that record's prediction away from the overall average, conditional on all the other features. Things get more complicated in the nonlinear case of a gradient-boosted machine or random forest. That is where SHAP values really add insight. SHAP values are extremely useful, so if this has piqued your interest, I highly suggest you read more about them (for instance, see Lundberg 2017).

So how can we use SHAP values in clustering? Well, because SHAP values deconstruct each individual prediction into the contribution from each feature, a data set with n observations and m features would generate a matrix of n x m SHAP values. This means that we have a new matrix that can effectively replace every feature in our data set with a new feature for our clustering model. Why is this interesting? Because it puts everything on a consistent interpretation and scale. It can also help us preprocess our data in desirable ways when dealing with categorical data, especially if the feature has high cardinality (in a way similar to impact encoding). Additionally, to the extent that the supervised model's response corresponds to the types of behaviors you wish to use your cluster model on, it could result in tighter clusters.

The clear challenge with this approach is the requirement to first build a supervised model from which to get the SHAP values. This is a luxury that we often don't have, but if we do, we should exploit it.

SEMI-SUPERVISED CLUSTERING

There has been promising research on incorporating outside knowledge into the clustering process. Semi-supervised clustering is an approach that can be used either when we have partially labeled data or when we want to enforce prior knowledge into the model. Typically, this prior knowledge is represented by specifying a constraint that two observations must be in the same cluster or conversely that two observations can't be in the same cluster. Semi-supervised clustering can be particularly useful when labeling instances is expensive or when you have a strong desire to enforce constraints based upon the business context of the model. How this works is that the algorithm takes user-provided input in terms of labels or constraints and accounts for it in the model-building process. As an example, imagine building a clustering model using age bands. You might want to enforce a constraint that records with younger age bands should not be in the same cluster as records with older age bands while still desiring to use age as a way to measure similarity between records.

CONCLUSION

In many clustering problems, simple approaches like k-means can provide "good enough" results, and in my anecdotal experience they often provide strong baselines that can be hard to improve upon. However, by digging a little deeper you may find ways to take your results to the next level.

Although clustering is not always as clear-cut as classification or regression problems, it doesn't mean that all answers are equally good. More often than not, in the real world you will be creating clusters as a tool to achieve a goal, and a good clustering model should be measured by how well it helps you achieve that goal. There may not be a way to unambiguously measure whether or not you have "the best" model, but the end goal should inform all your modeling decisions. ■

Michael Niemerg, FSA, MAAA, is predictive modeling manager at Milliman IntelliScript. He can be contacted at *michael.niemerg@milliman.com*.

BIBLIOGRAPHY

Aggarwal C., Reddy C. Data Clustering: Algorithms and Applications, 1st Edition (2013).

Awasthi, P., Zadeh, R.B., Supervised Clustering (2010). NIPS'10 Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1.

Bair, E., Semi-supervised clustering methods (2013). Wiley Interdisciplinary Reviews: Computational Statistics 5(5):349-361.

Lundberg S., Lee S., A unified approach to interpreting model predictions (2017). Advances in Neural Information Processing Systems, 4765-4774.