Clustering via Model Bias

Jesse Lovitt

Abstract

Clustering techniques aim to group data placing similar items in groups together. There are a wide variety of techniques available for clustering each with its own trade offs [Rokach, Lior, and Oded Maimon 2016]. Each of these clustering methods has a different formulation of what items should go together in a group. Thus there is no clear best method for clustering, and even given a set of fixed data there may not be a clear best way to cluster it. This fact gives us a continual need for more variety in the types of clustering that can be done. Work done in deep neural networks shows that they can be effectively used for feature learning. Which features of our data are taken into account during clustering obviously will have a large effect on the clustering outcome. This work investigates the relationship between deep convolutional neural networks and the concept of inter-item similarity for clustering. Specifically, a set of underparameterized deep convolutional autoencoders is used to cluster data from the MNIST digits data set.

Introduction

Variety of clustering Methods

Clustering of data is an important technique in data analysis and machine learning. This primarily unsupervised technique aims at creating clusters of similar data. The concept of similar however can be very data or situation dependent. As an example, consider the task of clustering image data. One might hope to cluster images by dominant color, by style, by subject matter, by brightness, etc… The list of possibilities is endless. [Rokach, Lior, and Oded Maimon 2016] List 9 different general similarity metrics in their survey of clustering techniques. Each of these similarity measures has its place depending on the type of data being clustered and the desired outcome of the clustering. This data and situation dependency gives rise to a wide variety of clustering algorithms and variations therein. Even with the wide variety that already exists there is a continual need for more just as there is a continual supply of myriad data sources and clustering intents.

Big Categories

The vast number of clustering methods can be themselves clustered into Partitioning, hierarchical, Density based, Model based, and Grid based themes. Each of these considers the concept of similarity differently. A grid based scheme for example will split the feature space of data into a grid in order to create clusters. In the resulting grid there is a similarity among all members of a grid cell or cluster. In other words no two members of a grid cell or cluster should be dissimilar. There exists an all to all similarity within the cluster. Density based methods however, group data items so that any item in a cluster must be similar to some subset of other items in that cluster, regardless of how dissimilar it is to other items in the same cluster but not in that subset. For this type there is a one to some similarity within a cluster. Regardless of the differences in algorithm used, the concept of similarity is in use in some way.

Categories of Interest to This Work

The work described here is a mix between a model based approach and a partitioning approach.

In a partitioning approach, the data is split into K groups, with the K usually being provided by the user. The data examples are then each considered for migration to a different cluster in order to improve the clustering quality. The quality of the clustering is often times measured by the intra-cluster similarity and the inter-cluster difference. Examples of such algorithms are K-means, K-medoids, and K-prototypes [Rokach, Lior, and Oded Maimon 2016]. All of these algorithms are updated in a respect similar to Expectation Maximization [Moon, Tood 1996]. A brief overview of the process is as follows. To start the process an initial definition of each cluster is derived. This definition is often in the feature space of the examples and can be simply a randomly chosen data example per cluster. In the first phase each example is then assigned to the cluster that is most similar to it. The second phase updates the cluster definition to better reflect the examples assigned to it. This new cluster definition often results in a change in the cluster to example similarities and thus the opportunity for a better set of assignments from examples to clusters. The two phases are repeated until there are no changes in the cluster definitions or the set of example to cluster assignments. The model bias clustering performed here follows a similar algorithm.

In model based clustering, a fit between a model and the data is attempted. Upon success, the model then explains the clustering of the data. Kohonen's self organizing maps [Kohonen 1990] is an example of such an algorithm. In this example, a set of Neurons are arranged in a grid with each connected to its neighbors via a weight. Each neuron is assigned a unit vector in the data space. As the clustering progresses, each data element is assigned to a neuron in a winner takes all competition based on a similarity between the neuron's assigned vector and the data vector. Each neuron then updates it's own vector based on it's assigned data and it's neighbor's values. In the end each neuron defines a cluster and the data assigned to it, the members of that cluster. Though this example is very similar to a partitioning scheme with the winner takes all assignment and the neuron vector updates, it is subtly different. There is an assumed model that persists throughout the clustering process that adds a bias toward a particular type of clustering. In the self organizing maps, a given neuron A's neighbors will update their own vectors to be more similar to A's own, even though the data examples assigned to A are not assigned to the neighbors. This sort of aggressive competition introduced by the model biases and drives the clustering toward different solutions than a partitioning scheme like K-means would.

Feature Selection

The concept of Similarity pervades all of the clustering methods described. The process of determining similarity however requires that one have access to some features of the data in order to assess two examples' similarity. In clustering, feature selection plays a very important role. One may describe the ocean and the sky as similar when using color as a feature, yet very different if using density as a feature. Algorithms such as K-means are not capable of determining valuable features from useless features and is susceptible to having useful features overwhelmed by useless ones if the quantities of meaningless features are high. In many cases it is important to use the knowledge of someone who is an expert in the field originating the data to sift through features and identify those of interest. This can however be time consuming and such experts are not always available.

Deep Feature Learning

Deep learning techniques have recently shown impressive performance in both classification and feature learning. Using these techniques it is possible to learn feature sets that are tailored to represent the given data. These techniques are not only in some cases relieving us of the burden of crafting features via domain expert knowledge, but in some cases they derive features that lead to performances that are better than those hand crafted ones can give. [Jarrett, Kevin, et al 2009]. In addition the features that are learned are hierarchical. This hierarchical feature learning is important because it allows us to represent a data item with an abstracted feature set. The importance of this is most evident in image data. The correlation between any given pixel value and a concept contained in an image is almost always near zero. Pixel level features are not particularly useful. It is in their relationships in large groups that they become valuable. Hierarchical features take into account the interrelations of a large number of pixels to form a single feature. It is precisely this fact that makes hierarchical features so important in image processing. Because deep learning can build abstracted features that are useful for representing image content and remove the dependence on mostly informationless pixel level data, it seems that it would be useful then to consider these hierarchical features when computing image similarity during clustering.

Model Bias

All machine learning models have some sort of bias. This bias restricts the modeling capability of the algorithm somewhat but allows it to generalize its knowledge to unseen data. In the context of a deep neural network, this bias takes the form of the types of features the network has learned to represent. This bias can be driven by the network architecture itself such as using convolution layers, network in network [Lin, Min et al. 2013], or LSTM units [Hochreiter, Sepp, and Jürgen Schmidhuber 1997]. The bias can be driven by simply the number of hidden units or the connectivity patterns among them. Corruption techniques such a dropout or dropconnect introduce bias into the model [Srivastava, Nitish, et al. 2014] [Wan, Li, et al. 2013]. Similarly adding noise to the data can bias the model [Vincent, Pascal, et al. 2008]. In every case the bias in a model makes the model better at representing certain data, and poorer at representing other data.

Autoencoder

An autoencoder is a multilayer neural network that can conceptually be split into two halves, and encoder and a decoder. The Encoder translates an input data into a representation in the space of its hidden neuron's. The decoder then takes the hidden representation of the data from the encoder and reproduces the original data. A variation on this is the convolutional autoencoder[Masci, Jonathan, et al. 2011]. This type of autoencoder simply uses convolutional layers in its architecture. The depth of an autoencoder is limited only by the same limitations as any other neural network. Thus, we can build a deep autoencoder. We can then think of the encoder half of the autoencoder as a feature learner. This feature learner is subject to the same methods of introducing bias as any other deep network.

Using Model Bias as a Cluster Attractor

This work explores what clustering results are achieved when the definition of similarity between examples is derived from how well they fit a particular model's bias. In particular I investigate a partitioning clustering scheme where each cluster is defined by a deep convolutional autoencoder. The concept of similarity between a data example and a cluster then becomes the inverse of the error with which the cluster autoencoder reproduces the data example.

The autoencoders used are underparameterized and fed data corrupted with noise such that they are incapable of modeling the whole data set to any reasonable degree, but could potentially model a single class from the data set. Clusters are formed by grouping those examples together that are best represented by a given autoencoder.

A particular question of interest is whether these deep models can cluster images by subject matter. The work by [Szegedy, Christian, et al. 2015] on GoogLenet claims that the subject matter class can be obtained simply by observing the highest level convolution layer. This suggests that the features learned in a deep enough network correspond to subject matter labels. If this is true and an underparameterized autoencoder can only learn a limited set of features, it stands to reason that such an autoencoder would also be limited to modeling a limited set of image subject matter's. A clustering based on these autoencoders as cluster attractors would then be able to cluster by subject matter.

Bear in mind that this hypothesis is not necessarily mathematically sound. It is simply a hunch who's value will be born out by the data the experiment produces.

Methods

The autoencoder architecture is a central piece of this work as it is this architecture that will drive the autoencoder bias and thus the cluster definitions.

Corruption

For the MNIST data set each autoencoder's first layer is a corruption layer. This layer stochastically masks out 20% of the image pixels. Because the MNIST digits are black and white images, this has the effect of making 20% of the white written digit pixels black. This effect helps generalize the autoencoder model so that it does not simply memorize specific examples. [Vincent, Pascal, et al. 2008]

The encoder half of the autoencoder was then continued with L convolution layers. This number is varied for different trainings to investigate the effect of different depths of hierarchical features on cluster performance. The parameters for each of these layers was set by hand to achieve a highly biased model, still capable of modeling a class. Initially the autoencoder was trained in a greedy layer

Fully Connected Layer

Each encoder was completed with a final fully connected layer. The decoder half of the model consisted of exactly the fully connected and convolution layers of the encoder with identical parameters only sequenced in reverse order. The weights of the decoder were independent of those of the encoder.

Parameter Selection

The parameters for the autoencoder had to be chosen to maximize bias, yet still allow modeling of a single class. To do this the model was trained in a greedy layer-wise fashion on a set of 8 examples from a single class. At each layer, the number of hidden neurons was adjusted so that the reconstruction was still recognizable but of degraded quality. An attempt was also made to ensure that the reconstruction loss was consistent throughout the greedy training so that one layer did not model the data any better than any other layer. As can be seen from the table of model parameters, this process yielded different parameterizations for similar layers in different trainings. To be practical, more work would need to be done to make the process of parameterizing a biased autoencoder easier and less subjective.

Pretraining

Once parameterization was completed two stages of pretraining were completed. The first stage had a single autoencoder trained greedily layer-wise on all the data. This yielded poor reconstruction and high error but served as a quick way to initialize weights. The second stage replicated this model 10 times. Each of these replicas was then trained on it's own set of 8 homogeneously labeled examples. This step served to bias the model toward a single class in the data. Although this step deviates from the spirit of unsupervised clustering, the burden of producing merely 8 examples per label is quite small.

Clustering

Once the 10 models had been trained on their respective labeled sets, they then were assigned as cluster anchors in the partition clustering algorithm. The clustering is an iterative process that proceeds much like Expectation Maximization. It begins by feeding all of the data into each of the 10 anchors. Each of the data items is then mapped to the anchor that encodes it and reproduces it with the least error. Each of the autoencoders is then trained for a fixed number of steps on only the data that is mapped to it. From here the process repeats until a set percentage of the data items do not change their mapping.

Algorithm

The general clustering algorithm is as follows:

```
map_0 ← { (d,null) } for all d in data
t ← 0
n ← n_0

Do
  t += 1
  #Expectation
  map_t ← {}
  For d in data
    For a in anchors
      c ← encode d in a
      r ← decode c in a
      e_a ← abs(d − r)
    map_t.add( (d, argmin_a(e)) )
  #Maximization
  For a in anchors
    For n steps
      Train a on all d such that (d,a) exists in map_t
  n += Δn
While( map_t-1 != map_t)
```

Evaluation

The MNIST hand written digit data set used contains a human generated label for each image where the labels indicate which numeral is displayed [Yann Lecun, Corinna Cortes.]. The CIFAR-10 natural image data set contains cropped and centered images of 10 distinct natural items such as horses, cars, dogs, cats, etc [Krizhevsky, Alex, and Geoffrey Hinton.

2009]. The labels here indicate which type of item is the centered object in the image. In both cases the labels categorize the image's subject matter. These subject matter labels serve as an evaluation measure to determine the extent to which the autoencoder bias based clustering was effective in grouping images of like subject matter.

The clustering technique is evaluated using two measures of entropy and by the measure of accuracy. The first entropy that is computed is that of the clusters. A separate entropy is computed for each cluster. To do this a count is obtained for every true label assigned to the cluster. Call this set of counts C. The entropy is obtained by $entropy = \sum_{c \in C} \frac{c}{s} \log \frac{c}{s}$ where s is the number of examples assigned to the cluster. A cluster that has only a single label assigned to it would have an entropy of 0. A cluster that is homogeneous in the labels assigned to it would have an entropy of $\log_2 \frac{1}{L}$ where L is the number of labels. The entropy of the label assignments is also computed. This follows the same process as above with the role of labels and columns interchanged.

The accuracy is computed by fixing a class label to each column. This assigned class label is simply the majority label of those labels mapped to it. All examples mapped to the column that are not of this majority class are counted as misclassified. This misclassification count is computed for each column and the misclassified counts summed. The accuracy is then $1 - \frac{misclassified}{n}$, with n being the total number of data examples.

Results

MNIST

Clustering of the MNIST data was performed with 4 different networks of varying depth. The network parameters can be seen in Table 1. For each configuration, the decoder was identical to the encoder but in reversed order.

Network Parameters

Depth	Corruption Layer	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7
2	20.00%	Convolutional: 2 5x5 @ 2	Fully Connected: 4					
4	20.00%	Convolutional: 4 5x5 @ 2	Convolutional: 8 5x5 @ 2	Convolutional: 16 3x3@ 1	Fully Connected: 8			
5	20.00%	Convolutional: 4 5x5 @ 2	Convolutional: 8 5x5 @ 2	Convolutional: 12 3x3@ 1	Convolutional: 16 3x3@ 1	Fully Connected: 4		
6	20.00%	Convolutional: 4 5x5 @ 2	Convolutional: 6 5x5 @ 2	Convolutional: 8 3x3@ 1	Convolutional: 16 3x3@ 1	Fully Connected: 32	Fully Connected: 4	
7	20.00%	Convolutional: 2 5x5 @ 2	Convolutional: 4 5x5 @ 2	Convolutional: 6 3x3@ 1	Convolutional: 18 3x3@ 1	Fully Connected: 64	Fully Connected: 16	Fully Connected: 4

Table 1: Autoencoder Parameterization

The accuracies obtained among the different network architectures are quite similar as can be seen in Table 2 .

Table 2: Accuracies **Accuracy per Network Depth**

Depth	2	4	5	6	7
Accuracy	0.791	0.710	0.769	0.752	0.76

Interestingly the most constrained network, the two layer, cluster's with the highest accuracy for the given class labels. The network with the least constrained version of the layers used, the four layer, cluster's with the lowest accuracy. These results are not competitive with [Bühler, Thomas, and Matthias Hein. 2009] who achieved 87.1% accuracy on MNIST using the same criteria and a spectral graph laplacian clustering algorithm. However, as an initial look at the idea of model bias based clustering the results here are in the ballpark and could undoubtedly be improved upon.

The performance of the two layer architecture follows. The means of the examples assigned to each anchor after convergence are shown in Illustration 1. These are obtained by simply finding the per pixel mean value over all of the images assigned to each anchor. The series of images shown in Illustration 1 are the means for each of the 10 anchors where the anchors are ordered by their majority class.

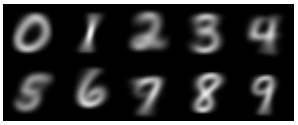

Illustration 1: Mean Image per Anchor

The label content of the examples mapped to each class as well as the entropies can be seen in Table 3. The anchor column numbers have been assigned according to the majority class label. In this way the inner content of the table can be read like a confusion matrix. This interpretation shows the big problem areas in terms of class label accuracy being “9” digits being mapped to the “4” column and “5” digits being assigned to the column dominant in “3”s. The class assignment entropy is shown in the right margin of Table 3. This can be interpreted as a quality measure of how well a specific class label was modeled. An entropy of 0 indicates that the class label was modeled solely by a single anchor column, and a high value (max of 3.32) indicates that the examples of that class label are distributed evenly among the anchors. Similarly the anchor assignment entropies are shown in the bottom margin of Table 3. This score can be interpreted as indicating how homogeneous an anchor is in the labels assigned to it. Here again, 0 entropy indicates a column that has only a single label type assigned to it and 3.32 indicates an anchor with an equal number of each label type.

2 layer net

True Class \ Column #	0	1	2	3	4	5	6	7	8	9	
Class: 0	5309	1	174	25	5	270	84	2	51	0	Entropy: 0.476
Class: 1	0	6329	43	5	10	116	27	13	178	18	Entropy: 0.322
Class: 2	31	16	5482	35	51	94	37	54	151	1	Entropy: 0.424
Class: 3	10	27	415	4689	84	244	20	65	564	10	Entropy: 0.905
Class: 4	1	8	123	1	4704	104	121	316	40	423	Entropy: 0.801
Class: 5	32	3	298	1174	176	3285	94	24	322	10	Entropy: 1.213
Class: 6	38	24	704	12	7	124	4989	0	15	0	Entropy: 0.568
Class: 7	16	33	95	0	592	31	0	5223	65	208	Entropy: 0.667
Class: 8	6	31	528	426	114	508	24	51	4150	10	Entropy: 1.049
Class: 9	15	10	78	71	3290	33	4	738	85	1621	Entropy: 1.170
	Entropy: 0.174	Entropy: 0.157	Entropy: 1.203	Entropy: 0.839	Entropy: 1.108	Entropy: 1.221	Entropy: 0.404	Entropy: 0.728	Entropy: 1.033	Entropy: 0.887	

Table 3: 2 Layer Network Performance

CIFAR-10

The results for the CIFAR-10 data set were less successful from the standpoint of recovering class labels. A two layer encoder network was used for each anchor with a final accuracy of 25.38% It can be seen from Illustration 3 and Illustration 4 that the cluster assignments are dominated by the image backgrounds in many cases. This is likely partly a

function of the error computation. Because the error is computed on a per pixel basis, the large number of pixels taken up by the image's background makes it a dominant force in the clustering.

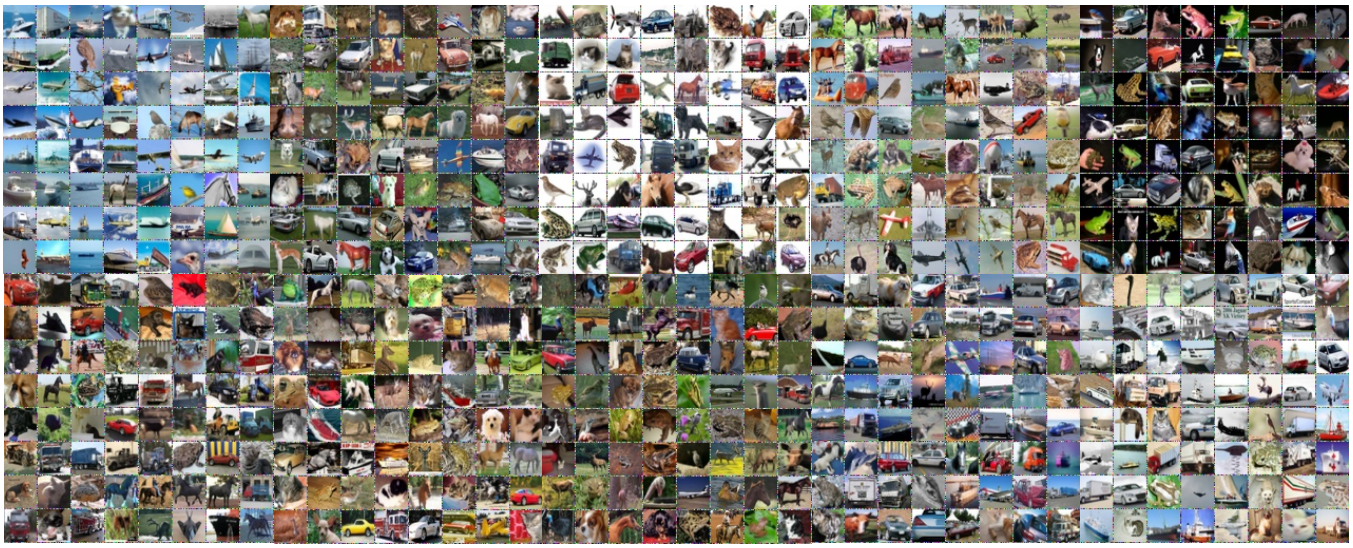

Illustration 2: Exemplar Images per Anchor

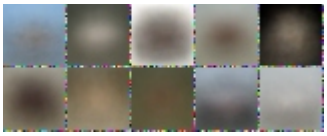

Illustration 3: Mean Image per Anchor

Future Work

One of the prominent observations in terms of CIFAR-10 is the dominance of the background color in the clustering. This effect prevents the recovery of class of the subject matter of the image since this subject matter takes up fewer of the image pixels. There are a couple of avenues of investigation regarding this.

Part of the dominance of the background color in the clustering is undoubtedly a product of the per pixel error error being used. Abandoning the per pixel error in favor of a more holistic error metric would be beneficial in this respect. In particular the work by [Goodfellow, Ian, et al. 2014] using a separate neural network to evaluate the authenticity and content of an autoencoder reconstruction seems promising.

Another strategy might be to use ZCA whitening [Krizhevsky, Alex, and Geoffrey Hinton. 2009] or difference of gaussians on the images before training to eliminate the importance of large areas of uniform color and give more weight to the areas where the image has variation.

One other area where more work would be needed is the parameterization of the autoencoders. Currently this is a trial and error sort of process, where it is unclear what type of outcome will result from a given parameterization.

Bibliography

Bühler, Thomas, and Matthias Hein. "Spectral clustering based on the graph p-Laplacian." *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009.

Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in Neural Information Processing Systems*. 2014.

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

Jarrett, Kevin, et al. "What is the best multi-stage architecture for object recognition?." *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009.

Kohonen, Teuvo. "The self-organizing map." *Proceedings of the IEEE* 78.9 (1990): 1464-1480.

Krizhevsky, Alex, and Geoffrey Hinton. "Learning multiple layers of features from tiny images." (2009).

Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *arXiv preprint arXiv:1312.4400* (2013).

Masci, Jonathan, et al. "Stacked convolutional auto-encoders for hierarchical feature extraction." *Artificial Neural Networks and Machine Learning–ICANN 2011*. Springer Berlin Heidelberg, 2011. 52-59.

Moon, Tood K. "The expectation-maximization algorithm." *Signal processing magazine, IEEE* 13.6 (1996): 47-60.

Rokach, Lior, and Oded Maimon. “Clustering Methods.” *Data Mining and Knowledge Discovery Handbook*. Springer, 2005. 321–352. *Google Scholar*. Web. 19 Apr. 2016.

Srivastava, Nitish, et al. "Dropout: A simple way to prevent neural networks from overfitting." *The Journal of Machine Learning Research* 15.1 (2014): 1929-1958.

Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." *Proceedings of the 25th international conference on Machine learning*. ACM, 2008.

Wan, Li, et al. "Regularization of neural networks using dropconnect." *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. 2013.

Yann Lecun, Corinna Cortes. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/