ECE571 Pattern Recognition

Project3: Color Image Compression Using Unsupervised Learning

Yang Xu

yxu71@vols.utk.edu

Abstract

Different from supervised learning, unsupervised learning deals with problems without any prior knowledge. The learner tries to self-organize or find the best way to model the inputs [1]. Clustering is one common application in unsupervised learning, which has been used for data mining, data compression, etc. Although there are multiple clustering algorithms that are available nowadays, it's still be tricky to apply them to a certain dataset. Many issues must be considered, for instance computation complexity. This project will implement three clustering algorithms, K-means, winner-take-all, and Kohonen maps respectively, and use these three algorithms to perform image compression task.

Introduction

K-means | K-means clustering is one very common clustering method. It is also one clustering algorithm developed in very early stage, and it is broadly applied in many fields. In computer vision, K-means clustering can efficiently perform segmentation and compression. Assuming there are k clusters within the dataset, the algorithm learns to partition hyperspace into k regions and each data point is assigned a class label based on its location in the partitioned space [2]. The first step of K-means is vectorization. In this project, vectorization can be thought as converting the RGB pixel value into a 3-dimension vector. This vector is also feature of the pixel. Another critical step in this algorithm is initialization of cluster centroids. It's critical because a "bad" initialization can lead to local optimum and it turns out an unideal clustering result.

To update centroids, each data point will be first assigned a class by finding the closest cluster centroid. Once all data points are classified, centroids will be assigned the new means of classes. This process will stop when no class label is changed. Back to the issue of reaching to local optimum. Supposing one centroid is initialized close to one extreme data point or an outlier, this centroid won't be able to move out the local minimum caused by this extreme or outlier.

Winner-take-all | Given points without labels, winner-take-all algorithm is learning to find representatives. These representatives are also called prototypes [3]. In clustering, data points will be derivatives of these prototypes. Different from K-means, winner-take-all can start with or without initialization. Without initialization, prototypes can start with sub samples in the first iteration, and then move towards the optimum in the coming iterations. The learning process has no large difference from K-means. However, no data points are assigned labels while learning. Therefore, prototypes won't stop moving until stop is indicated. This property also addresses local minimum to some degree, because continously feeding training will push prototypes to move out local minimum.

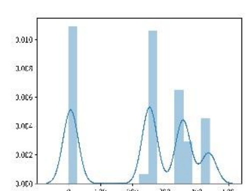
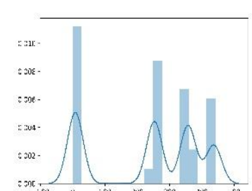
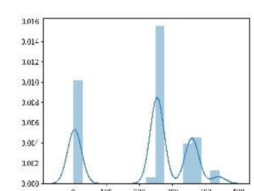
Kohonen maps | Kohonen map is called self-organizing map, which can be regarded as an extension of winner-take-all net [4, 5]. But different from winner-take-all, Kohonen map considers this map as a connected net. Since it is a connected net, when one node is updated, the connected nodes will also be updated. Given enough iterations, this net will organize it self as its name suggests. To connect the net, we can introduce an influence function to define how nodes are affected by their neighboring nodes. Furthermore, prototype is replaced by best matching unit (BMU). BMU is the node that is the closest to the input. Given one input, the net will organize itself around the BMU under the influence function. Similar to winner-take-all, kohonen map can be initialized with random weights for each node or without initialization by assigning all weights to zeros. However, zero start may decay the convergence of learning, because net has to first move all nodes out zero. Therefore, the good approach is random initialization.

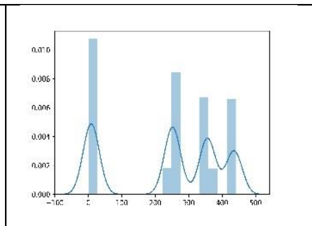
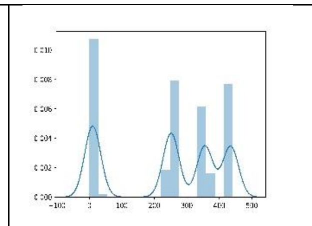
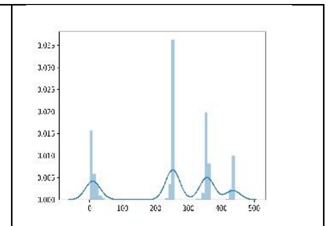
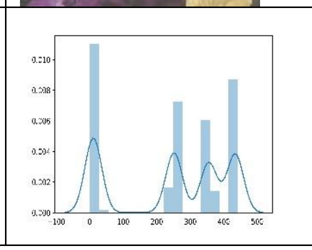
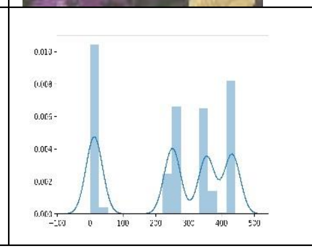
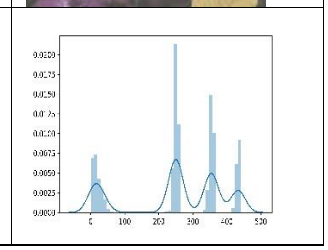
Technical approach

In this project, I compressed the flower image into 256, 64 and 32 colors respectively. Since winner-take-all and Kohonen maps are involved of more complicated design, I didn't implement these two algorithms as sophisticated as online python libraries. The main differences of winner-take-all and Kohonen maps from online sources is the learning rate and convergence (iteration). 1. Most sophisticated algorithms deal with learning rate with decaying. However, in this implementation, I set the learning rate as 0.01 with no decay as learning progresses. 2. When stops learning is another critical issue. I set iteration as one parameter in both winner-take-all and Kohonen maps functions. The algorithms will stop once iteration ends. However, most sophisticated online algorithms will stop once convergence. Considering the large mount of pixels in this image, I chose to randomly down samples to a training batch in each iteration. This greatly saves learning time and computation power. Meanwhile, the influence of BMU in Kohonen maps is defined by 5X5 gaussian kernel with 1 sigma.

To quantify quality loss, I simply calculated the Euclidean distance of the compressed pixel with original pixel, and then I plotted histograms to represent the overall quality loss after image compression.

Experiments and results

	**K-means**	**Winner-take-all**	**Kohonen maps**
256 colors			
Quality loss			
64 colors			

Quality loss			
32 colors			
Quality loss			

Discussion

The three clustering algorithms implemented in this project can compress images to a comparable resolution, respect to 256, 64 and 32 colors. It is very difficult to distinguish results from these three algorithms. However, evaluation of quality loss shows that both K-means and winner-take-all can keep a large proportion of pixels with less loss. On another hand, Kohonen maps avoid large loss of pixel. Furthermore, it seems that Kohonen maps compress image by converging to a “middle point” given enough iterations. This can be reflected from the quality histogram: there is a high spike in the middle.

To improve efficacy and sophistication of these three clustering algorithms, there are several breaking points that can be approached.

1. Initialization of cluster centroids can be achieved by assuming centroids in a multi-modal gaussian distribution. Therefore, it can avoid extreme centroids and reaching to local minimum to some degree.
2. Build-in of loss function in winner-take-all and Kohonen maps. The reason behind is finding a good stop point. Implementation of loss function while learning can be an indicator for stop point. The loss function can also present how good the learner reaches.
3. The third strategy is learning through batch. In this case, the image is 120X120, which is not a big photo for clustering task. However, most real-life cases deal with even larger images. Clustering based on the whole dataset would be time- and computation-consuming. Learning through batch or down-sampling can greatly reduce time of learning. If batch implemented, another issue would be how to make batch representable for the whole dataset, and it is also affected by the size of batch.

Appendix

All implementation is available in the python script and should be run by Python 3.

Reference

1. Hinton, Geoffrey E., Terrence Joseph Sejnowski, and Tomaso A. Poggio, eds. *Unsupervised learning: foundations of neural computation.* MIT press, 1999.
2. MacQueen, James. "Some methods for classification and analysis of multivariate observations." *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. No. 14. 1967.
3. Carpenter, Gail A., and Stephen Grossberg. "A massively parallel architecture for a self-organizing neural pattern recognition machine." *Computer vision, graphics, and image processing* 37.1 (1987): 54-115.
4. Kohonen, Teuvo. "Self-organized formation of topologically correct feature maps." *Biological cybernetics* 43.1 (1982): 59-69.
5. Kohonen, Teuvo, and Timo Honkela. "Kohonen network." *Scholarpedia* 2.1 (2007): 1568.