CS 374 Assignment #10
Unsupervised Learning

Due the week of May 3, 2021

—

This week you will have the opportunity to explore in detail two algorithms for unsupervised learning – one clustering algorithm and one autoencoding algorithm. In unsupervised learning, data are presented to an algorithm in the form of training examples. The difference between these training examples and those you have seen earlier this semester, is that they are unlabeled. That is, the training examples are described by attribute information, but they have no associated class. The goal in *clustering* is to find groups of examples that are similar to each other but distinct from other groups of examples. The goal in *autoencoding algorithms* is to infer new features from existing ones. This is particularly useful in the context of deep learning, which you will revisit this week.

This will be your last structured assignment. During the final weeks of the semester, you will work with your tutorial partner (though larger groups are possible) to construct your own assignments, which you will present in the final week of classes.

What to turn in. For this assignment, you will turn in an implementation of k-means. As is typical for coding assignments, I encourage you to work with your tutorial partner. Your other deliverable is a polished presentation of the Le et al. paper.

What to expect during the meeting. Your job during the meeting will be to present clustering algorithms (both k-means and EM, with a focus on the former), demonstrate your k-means implementation, and review the code. This should take approximately 30 minutes. You will then have 45 minutes to carefully present the paper. Note that the presentation is meant to focus on explaining the content of the paper. If we have time left after the presentation, we will proceed to a critical analysis of the work.

1 Clustering

This week you will consider the simple k-means and EM clustering algorithms, with a focus on the former.

The basic idea of the k-means algorithm is as follows. Randomly select k points in your example space. These are taken to be the initial centers of clusters. (So there will be k clusters.) The training instances are then assigned to a cluster, based on their distance from each of the k centers. Once all examples have been assigned to clusters, k new centers are found by computing the mean of each cluster. The process then repeats. All training instances are assigned to clusters based on their distances to the new centers, etc. Once the centers become stable, you're done.

There are clearly advantages and disadvantages to this algorithm. There are also alternatives, such as the EM algorithm. EM assumes that the training instances were generated by a mixture of Gaussians and uses this assumption to perform clustering in a manner roughly analogous to k-means. It hypothesizes the values of the parameters and then revises those hypotheses with each iteration of the algorithm.

1.1 Reading

There are many sources from which you can learn about k-means and EM. They include:

- Mitchell, Section 6.12,
- Alpaydin, Sections 7.1-7.4,
- Witten and Frank, pages 137-138, 262-266, 337-338.

You aren't required to read all of these. It's up to you to read as much as you need in order to do the implementation of k-means and to have a good high-level understanding of EM.

1.2 Implementation

This week one of your exercises will involve the implementation of the k-means algorithm. This algorithm is conceptually fairly simple, but as you've seen with other algorithms you've implemented this semester, some details can be tricky and debugging is non-trivial.

As the readings make clear, there is no known theoretical way to select an optimal value for k, the number of clusters. For this exercise, however, you will be training on data sets for which we have class labels. This, of course, is not what we'd expect when doing clustering, but the class information will allow you to see how k-means performs when it has a good value for k. *Though you will use the class information to choose a value for k, you should not treat the class labels as attributes!!!*

The data sets with which you will be working are `iris`, `glass`, and `vehicle`. I've selected these as they have only real-valued attributes. This is essential as you will need to compute the Euclidean distance from a training instance to the center of a cluster. (There are versions of clustering algorithms that work with discrete-valued attributes, but we won't consider them this week.) In order to handle real-valued data, you will undoubtedly need to make some changes to the code you wrote previously to read data from an ARFF file. The changes should be fairly small, however.

Your k-means clustering program should do the following:

- Read the training instances from the file. Don't discard the class information. While you can't use it for clustering, you will need it later for assigning names to the clusters and for checking the accuracy of the clusters. You do not need to normalize the attribute values.
- Apply the k-means algorithm to find clusters (e.g., three in the case of iris; four for vehicles).
- Assign each final cluster a name by choosing the most frequently occurring class label of examples in the cluster.
- Find the number of examples that were put in clusters in which they didn't belong. You can check your results by comparing with Weka. Note that your results for iris will likely be close to Weka's in most cases. The results in the other data sets will likely vary more.

2 Unsupervised Learning in Deep Learning

2.1 Reading

Please read

- "Building High-level Features Using Large Scale Unsupervised Learning", a paper by Le, Ranzato, Monga, Devin, Chen, Corrado, Dean, and Ng.

This paper appeared in the proceedings of ICML 2012, the 29th International Conference on Machine Learning. You can find a link to the paper by going to the Assignments page for this course. You will also find it on Glow.

2.2 Exercise

This is a great paper with a really cool result, but it isn't trivial to understand. You will likely need to consult other references. I recommend the *Deep Learning* text by Goodfellow, Bengio, and Courville. You can find it online at

`http://www.deeplearningbook.org`

Please be ready to present the Le et al. paper. You should be sure to cover:

- The motivation for the work.
- The algorithm.
- Experiment infrastructure and input data.
- Results.

You should plan to spend at least a third of the time on the algorithm. To do this well, you will need to provide more detail than the paper gives. Again, I suggest going back to the *Deep Learning* text. This is not to say that you have to present every detail of the algorithm, but I expect you to demonstrate a real grasp of at least the autoencoding component of the overall algorithm and a general understanding of the rest.

You may work with your tutorial partner(s) on a single presentation. Alternatively, simply coordinate the high-level structure of the presentation so that it flows smoothly. Note, however, that all students should be comfortable discussing the algorithm.