CS 6362: Machine Learning

Assignment 4

K-means and EM algorithms

Due: 11/19/2019, midnight

Total: 100 Points

1. Programming (80 points)

In this assignment you will implement a variant of the *K*-means clustering algorithm called λ-means clustering.

1.1 λ-Means

The *K*-means clustering algorithm groups instances into *K* clusters. Each cluster is represented by a prototype vector μ_k, which represents the mean of the examples in that cluster. Cluster assignments are "hard," meaning that an instance can belong to only a single cluster at a time.

The *K*-means algorithm works by assigning instances to the cluster whose prototype μ_k has the smallest distance to the instance (based on, e.g. Euclidean distance). The mean vectors μ_k are then updated based on the new assignments of instances to clusters, and this process is repeated using an EM-style iterative algorithm.

A limitation of *K*-means is that we must choose the number of clusters *K* in advance, which can be difficult to choose in practice. There is a variant of *K*-means, which we call λ-means, in which the number of clusters can change as the algorithm proceeds. This algorithm is very similar to *K*-means, with one difference: when assigning an instance $\mathbf{x}_i$ to a cluster, we will assign it to the lowest-distance cluster **unless** all of the clusters have a distance larger than some threshold λ. In this case, we then assign $\mathbf{x}_i$ to a new cluster, which is denoted cluster $K + 1$ if we previously had *K* clusters. The prototype vector for the new cluster will simply be the same vector as the instance, $\mu_{K+1} = \mathbf{x}_i$. The idea is that if an instance is not similar to any of the clusters, we should start a new one[1]

Clustering as a `Predictor`

You will implement two methods: *train* and *predict*. In unsupervised learning, the learner does not have access to labeled data. Your implementation should include the following functionality:

- The *train* method should learn the cluster parameters from the data. The result of the train method are the cluster parameters: the means μ_k and the number of clusters *K*.

[1] This algorithm (called "DP-means" here) is described in: B. Kulis and M.I. Jordan. Revisiting k-means: New Algorithms via Bayesian Nonparametrics. 29th International Conference on Machine Learning (ICML), 2012. (https://arxiv.org/abs/1111.0352)

- The *predict* method should label the examples by assigning them to the closest cluster. The value of the label should be the cluster index, i.e., an example closest to k^{th} cluster should receive label k.

1.2 Inference with EM

The *K*-means algorithm and the λ-means algorithm are based on an EM-style iterative algorithm. On each iteration, the algorithm computes 1 E-step and 1 M-step. In the E-step, cluster assignments r_{ik} are determined based on the current model parameters μ_k. r_{ik} is an **indicator** variable that is 1 if the i^{th} instance belongs to cluster k and 0 otherwise.

Suppose there are currently K clusters. You should set the indicator variable for these clusters as follows. For $k \in \{1, \ldots, K\}$:

$$r_{ik} = \begin{cases} 1 & \text{if } k = \arg min_j \left\|\boldsymbol{x}_i - \boldsymbol{\mu_j}\right\|^2 \text{ and } min_j \left\|\boldsymbol{x}_i - \boldsymbol{\mu_j}\right\|^2 \leq \lambda \\ 0 & otherwise \end{cases}$$

where $\|\boldsymbol{x} - \boldsymbol{x}'\|^2$ denotes the squared Euclidean distance, $\sum_{f=1}^{m}\left(x_f - x'_f\right)^2$. When computing this distance, assume that if a feature is not present in an instance **x**, then it has value 0.

Additionally, you must consider the possibility that the instance $\mathbf{x}_n$ is assigned to a new cluster, $K + 1$. The indicator for this is:

$$r_{n,K+1} = \begin{cases} 1 & \text{if } min_j \left\|\boldsymbol{x}_i - \boldsymbol{\mu_j}\right\|^2 > \lambda \\ 0 & otherwise \end{cases}$$

If $r_{n,K+1} = 1$, you should immediately set $\mu_{K+1} = \mathbf{x}_n$ and $K = K + 1$, before moving on to the next instance $\mathbf{x}_{n+1}$. You should not wait until the M-step to update μ_{K+1}.

Be sure to **iterate through the instances in the order they appear** in the dataset.

After the E-step, you will update the mean vectors μ_k for each cluster $k \in \{1,2, \ldots, K\}$

(where K may have increased during the E-step). This forms the M-step, in which the model parameters μ_k are updated using the new cluster assignments:

$$\mu_k = \frac{\sum_i r_{ik} x_i}{\sum_i r_{ik}}$$

This process will be repeated for a given number of iterations (see 1.5).

1.3 Cluster Initialization

As we discussed in class, different initializations will lead to different clustering solutions. In *K*-means, the standard initialization method is to randomly place each instance into one of the *K* clusters. However, in λ-means you can simply initialize the algorithm with only one cluster, i.e. $K = 1$. You should initialize the prototype vector to the mean of all instances: $\mu_1 = \bar{x} = \frac{1}{m}\sum_i x_i$

1.4 Lambda Value

Choose the default value of λ to be the max distance between points in the training data:

$$\lambda^{default} = \max_{i,j} \left(\sum_{i,j=1}^{m} \|x_i - x_j\|^2 \right)^{1/2}$$

Successively decrease λ by reasonable increments, and plot the values of λ against the average within cluster variance of the data. As λ decreases, the number of clusters should increase, and the average within cluster variance should decrease. Use a heuristic corresponding to the Occam's razor principle (look for the knee of the variance versus λ curve) to choose the appropriate number of clusters for this data set. We highly encourage you to experiment with different values of λ and see how it effects the number of clusters which are produced. Report the nature of the curve and discuss the results.

Bonus: There are more systematic ways to derive λ. For example, look up the paper, Comiter, M. Z., Cha, M., Kung, H. T., & Teerapittayanon, S. (2016). Lambda means clustering: automatic parameter search and distributed computing implementation. (https://dash.harvard.edu/handle/1/34390104). See if you can interpret and apply the method discussed in this paper, and see if it matches your empirical results. Discuss the two results.

1.5 Implementation Details

1.5.1 Tie-Breaking

When assigning an instance to a cluster, you may encounter ties, where the instance is equidistant to two or more clusters. In case of a tie, select the cluster with the lowest cluster index.

1.5.2 Empty Clusters

It is possible (although unlikely) that a cluster can become empty during the course of the algorithm. That is, there may exist a cluster $k\ such$ that $r_{ik} = 0, \quad i$. In this case, you should set $\mu_k = \mathbf{0}$. Do not remove empty clusters.

1.5.3 Expanding the Cluster Set

In standard K-means, you typically create arrays of size K to store the parameters. Since the number of clusters can grow with λ-means, you will need to set up data structures that can accommodate this. Find an effective way to implement this.

Evaluation

We will evaluate your implementation on two data sets that you can download from the links provided below.

1. Data set 1 is a High Dimensional Gaussian Data set (http://cs.joensuu.fi/sipu/datasets/), with $N = 1024$ data objects; $number\ of\ features = 32$; and $K = 16$ is the number of classes in the data. The data is numeric, and can be read from a text file.

2. The second data set is a water treatment plant data set that comes from the UCI repository: https://archive.ics.uci.edu/ml/datasets/Water+Treatment+Plant. This dataset comes from the daily measures of sensors in a urban waste water treatment plant. The objective is to classify the operational state of the plant in order to predict faults through the state variables of the plant at each of the stages of the treatment process. This domain has been stated as an ill-structured domain.

1.7 Analytical written assessment (20 points)

1) *K*-means and Overfitting Suppose we apply the *K*-means algorithm to a data set of m examples, x_1, ..., x_m. The *K*-means algorithm aims to partition the m observations into k sets ($k \leq m$) $S = S_1, S_2, \ldots, S_K$ so as to minimize the within-cluster sum of squares

$$\min_{S=\{S_1,S_2,\ldots,S_K\}} \sum_{k=1}^{K} \sum_{x_i \in S_k} \|x_i - \mu_k\|_2^2$$

where μ_k is the mean of points in S_k.

(a) Let γ_K denote the minimum of the above equation. Prove analytically that γ_K is non-increasing in K.

(b) We can also apply the kernel trick to the *K*-means algorithm. Suppose we have some mapping from the original space to the feature space, $x \rightarrow \varphi(x)$. Then we can rewrite the within cluster sum of squares equation as

$$\min_{S=\{S_1,S_2,\ldots,S_K\}} \sum_{k=1}^{K} \sum_{x_i \in S_k} \|\varphi(x_i) - \alpha_k\|_2^2$$

where α_k is the mean of points in S_k in the feature space, i.e,

$$\alpha_k = \frac{1}{|S_k|} \sum_{x_i \in S_k} \varphi(x_i)$$

Recall the kernel function is defined as $k(x, x_i) = \varphi(x) \cdot \varphi(x_i)$. Please derive the update formulas for this version of K-means.

Useful facts: $\|x - y\|_2^2 = (x - y) \cdot (x - y)$*;*

$(x + y) \cdot z = x \cdot z + y \cdot z$*; for a constant a,* $ax \cdot y = a(x \cdot y$

3. What to Submit

In each assignment you will submit two things.

1. **Code:** Your code as a zip file named assn3code.zip. **You must submit source**

code (python or java files). We will run your code so make sure it works ahead of time. Remember to submit all of the source code.

2. **Writeup:** Your writeup as a **PDF file** containing answers to the analytical questions asked in the assignment. Make sure to include your name in the writeup PDF.