CS 475 Machine Learning: Homework 4
Clustering, EM, Dimensionality Reduction, Graphical Models
Programming Assignment

Due: Wednesday April 15, 2020, 11:59pm

50 Points Total Version 1.0

Before you begin, please read the Homework 4 Introduction. The introduction has information on how to use the Python codebase, the data, and how to submit your assignment.

We have provided Python code to serve as a testbed for your algorithms. We'll use the same coding framework that we used in the first two homework assignments. You will fill in the details. Search for comments that begin with `TODO`; these sections need to be written. Your code for this assignment will all be within the `models.py` file; do not modify any of the other files.

In this assignment you will implement two variants of the K-means clustering algorithm, λ-means clustering and the stochastic K-means algorithm.

1 K-Means

The K-means clustering algorithm groups instances into K clusters. Each cluster is represented by a prototype vector $\boldsymbol{\mu}_k$, 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 $\boldsymbol{\mu}_k$ has the smallest distance to the instance (based on, for example, Euclidean distance). The mean vectors $\boldsymbol{\mu}_k$ 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 in practice.

2 λ-Means (25 points)

λ-means is a variant of K-means that addresses this issue, by changing the number of clusters as the algorithm proceeds. This algorithm is very similar to K-means but with one key difference: an instance $\mathbf{x}_i$ is assigned to the nearest existing cluster **unless** all of the cluster prototypes have a distance larger than some threshold λ. In that case, $\mathbf{x}_i$ is assigned to a new cluster (cluster $K+1$ if we previously had K clusters). The prototype vector for the new cluster is simply the same vector as the instance, $\boldsymbol{\mu}_{K+1} = \mathbf{x}_i$. The idea

here is that if an instance is not similar enough to any of the existing clusters, we should start a new one.[1]

2.1 Implementation

Your `LambdaMeans` class should inherit from `Model`, as you have done all semester. You will still implement the `fit` and `predict` methods. However, the behavior will be slightly different than in previous projects.

In unsupervised learning, the learner does not have access to labeled data. However, the datasets you have been using contain labels in the training data. You should not use these labels at all during learning. Additionally, since the clustering algorithm cannot read the correct labels, we must be clear about what "label" you are returning with `predict`.

Your implementation should include the following functionality:

- The `fit` method should learn the cluster parameters based on the training examples. While the labels are available in the provided data, the clustering algorithms should not use this information. The result of the `fit` method are the cluster parameters: the means $\boldsymbol{\mu}_k$ and the number of clusters K.
- 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 kth cluster should receive label k. While this document will describe the algorithm as if k is in the set $\{1, \ldots, K\}$, it is fine for your code to use the set $\{0, \ldots, K-1\}$ because the evaluation scripts (see Section 1.8) will give the same results whether you use 0-indexing or 1-indexing.

2.2 Inference with EM

The K-means algorithm and the λ-means algorithm are based on an EM-style iterative approach. On each iteration, the algorithm computes 1 E-step and 1 M-step.

2.2.1 E-Step

In the E-step, cluster assignments r_{nk} are determined based on the current model parameters $\boldsymbol{\mu}_k$. r_{nk} is an **indicator** variable that is 1 if the nth 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 $k \in \{1, \ldots, K\}$:

$$r_{nk} = \begin{cases} 1 & \text{if } \; k = \arg\min_j ||\mathbf{x}_n - \boldsymbol{\mu}_j||_2 \text{ and } \min_j ||\mathbf{x}_n - \boldsymbol{\mu}_j||_2 \leq \lambda \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $||\mathbf{x} - \mathbf{x}'||_2$ denotes the Euclidean distance, $\sqrt{\sum_{f=1}^{m}(x_f - x'_f)^2}$. When computing this distance, assume that if a feature is not present in an instance $\mathbf{x}$, 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:

[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.

$$r_{n,K+1} = \begin{cases} 1 & \text{if } \min_j ||\mathbf{x}_n - \boldsymbol{\mu}_j||_2 > \lambda \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

If $r_{n,K+1} = 1$, you should immediately set $\boldsymbol{\mu}_{K+1} = \mathbf{x}_n$ 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 $\boldsymbol{\mu}_{K+1}$.

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

2.2.2 M-Step

After the E-step, you will update the mean vectors $\boldsymbol{\mu}_k$ for each cluster $k \in \{1, \ldots, K\}$ (where K may have increased during the E-step). This forms the M-step, in which the model parameters $\boldsymbol{\mu}_k$ are updated using the new cluster assignments:

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\mathbf{x}_n}{\sum_n r_{nk}} \tag{3}$$

This process will be repeated for a given number of iterations. You can specify the number of training iterations using the command arg `--clustering-training-iterations`. The default number of iterations is 10.

2.3 Cluster Initialization

As we discussed in class, clustering objectives are non-convex. Therefore, different initializations will lead to different clustering solutions. In order for us to test the submitted code, we must have everyone use the same initialization method.

In K-means, a standard initialization method is to randomly place each instance into one of the K clusters. However, in λ-means you can initialize the algorithm with only one cluster ($K = 1$). You should initialize the prototype vector to the mean of all instances: $\boldsymbol{\mu}_1 = \bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n$.

2.4 Lambda Value

The default value of λ will be the average distance from each training instance to the mean of the training data:

$$\lambda^{(\text{default})} = \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{x}_i - \bar{\mathbf{x}}||_2, \tag{4}$$

where $\bar{\mathbf{x}}$ is the mean vector of the training instances (also the initialization of $\boldsymbol{\mu}_1$ in section 2.3).

We highly encourage you to experiment with different values of λ to see how it affects the number of clusters that are produced. In particular, think about how to choose λ in order to have K match the true number of classes in the dataset. You can adjust the value of λ with the command arg `--cluster-lambda`. The default value is 0.0, which indicates that you should use $\lambda^{(\text{default})}$.

2.5 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. One option is to simply use arrays of a fixed size, expanding the size of the array if you run out of space (by creating a larger array and copying the contents of the old array). Alternatively, you can expand the size of the data structure as you go along (though this will likely run slower). *Hint: see numpy's concatenate, hstack, and vstack functions.*

3 Stochastic K-means (25 points)

The Stochastic K-means algorithm (SKA) associates the data point $\mathbf{x}_i$ to a cluster based on a probability distribution that depends on the distance between the data point and the cluster centers. SKA is less dependent than K-means on the initial cluster choices. In SKA, we don't have a λ parameter since the number of clusters is fixed at initialization. You will use the command like arg `--number-of-clusters` to specify the value of K. The default number of clusters is 3.

3.1 Cluster Initialization

The cluster initialization is the same as that of the K-means algorithm; i.e. typically done randomly. However, to standardize results we require you to use the following clustering procedure:

- For $K = 1$, initialize by taking the mean of the data (as in Section 2.4).
- For larger K, choose centers that divide the range of each dimension into $K - 1$ equal pieces. For example:
  - For $K = 2$, choose one center with all of the minima and one center with all of the maxima.
  - For $K = 3$, choose a center with all maxima, one with all minima, and one where $x_{c,i} = (\max(x_i) + \min(x_i))/2,\ \forall i$.
  - For $K = 4$, choose the maxima, the minima, and prototypes at $x_{c1,i} = 1/3 * \max(x_i) + 2/3 * \min(x_i),\ \forall i$ and $x_{c2,i} = 2/3 * \max(x_i) + 1/3 * \min(x_i),\ \forall i$.
  - Continue this way for larger K.

3.2 Assigning data to clusters

At the beginning of each iteration, assign each data point $\mathbf{x}_n$ to the closest mean. You should set the indicator variable for the K clusters as follows. For $k \in \{1, \ldots, K\}$,

$$r_{nk} = \begin{cases} 1 & \text{if } \ k = \arg\min_j d(\mathbf{x}_n, \boldsymbol{\mu}_j) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The probability that $\mathbf{x}_n$ belongs to cluster k is defined as follows:

$$p_{nk}^{(i)} = \frac{\exp\left(\frac{-\beta^{(i)} d(x_n, \mu_k)}{\hat{d}(x_n)}\right)}{\sum_{j=1}^{K} \exp\left(\frac{-\beta^{(i)} d(x_n, \mu_j)}{\hat{d}(x_n)}\right)} \tag{6}$$

where $d(\mathbf{x}_n, \boldsymbol{\mu}_k)$ is the Euclidean distance between $\mathbf{x}_n$ and $\boldsymbol{\mu}_k$ and $\hat{d}(\mathbf{x}_n)$ is the mean Euclidean distance between $\mathbf{x}_n$ and all $\boldsymbol{\mu}_j$, $j \in \{1, \ldots, K\}$

$$\hat{d}(\mathbf{x}_n) = \frac{1}{K} \sum_{j=1}^{K} ||\mathbf{x}_n - \boldsymbol{\mu}_j||_2 \tag{7}$$

The model parameters μ_k should then be updated as

$$\boldsymbol{\mu}_k = \frac{\sum_n p_{nk} \mathbf{x}_n}{\sum_n p_{nk}} \tag{8}$$

3.3 How to choose β

The parameter β increases linearly with iteration, as follows:

$$\beta^{(i)} = c * i, \tag{9}$$

where i is the iteration number and c is a constant. You may want to experiment with different values of c, but use $c = 2$ in your final implementation. Note that in the limit, as $\beta \to \infty$, SKA behaves similarly to the K-means algorithm.

4 Implementation Details

These considerations apply to both λ-means and Stochastic K-means.

4.1 Tie-Breaking

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

4.2 Empty Clusters

It is possible (though unlikely) that a cluster can become empty while training the algorithm. That is, there may exist a cluster k such that $r_{nk} = 0, \ \forall n$. In this case, you should set $\boldsymbol{\mu}_k = \mathbf{0}$. Do not remove empty clusters.

5 Submitting

Please refer to Homework 4 Introduction for instructions on submitting your programming assignment.