CS 475 Machine Learning: Homework 6
K-means and EM algorithms
Due: Wednesday November 14, 2012, 12:00pm
100 Points Total Version 1.1

1 Programming (50 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]

You should create a class called `LambdaMeansPredictor`. The command line `algorithm` argument should be `lambda_means`.

1.1.1 Clustering as a `Predictor`

Your class should extend `Predictor`, as you have done all semester. You will still implement the `train` and `predict` methods. The behavior will be slightly different than usual, however.

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:

[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.

- The `train` 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 `train` method are the cluster parameters: the means μ_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 1.7) will give the same results whether you use 0-indexing or 1-indexing.

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_{nk} are determined based on the current model parameters μ_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 $k \in \{1, \ldots, K\}$:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j ||\mathbf{x}_n - \mu_j||^2 \text{ and } \min_j ||\mathbf{x}_n - \mu_j||^2 \leq \lambda \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

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:

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

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 $k \in \{1, \ldots, K\}$ (where K may have increased during the E-step). This is forms the M-step, in which the model parameters μ_k are updated using the new cluster assignments:

$$\mu_k = \frac{\sum_n r_{nk}\mathbf{x}_n}{\sum_n r_{nk}} \tag{3}$$

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

1.3 Cluster Initialization

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

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: $\mu_1 = \bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$.

1.4 Lambda Value

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

$$\lambda^{(\text{default})} = \frac{1}{N}\sum_{i=1}^{N}||\mathbf{x}_i - \bar{\mathbf{x}}||^2$$

where $\bar{\mathbf{x}}$ is the mean vector of the training instances (also the initialization of μ_1 in 1.3).

You *must* add a command line argument to allow this value to be adjusted via the command line: `cluster_lambda`. Add this command line option by adding the following code to the `createCommandLineOptions` method of `Classify`.

```
registerOption("cluster_lambda", "double", true, "The value of lambda in lambda-means.");
```

You can then read the value from the command line by adding the following to the main method of `Classify`:

```
double cluster_lambda = 0.0;
if (CommandLineUtilities.hasArg("cluster_lambda"))
    cluster_lambda = CommandLineUtilities.getOptionValueAsFloat("cluster_lambda");
```

How you implement the default value is up to you, but it probably makes more sense to set the default value of λ inside `LambdaMeansPredictor` rather than `Classify`. You can do this by setting the value to 0 by default (as in the above code example), then set the value to the default in Eq. 1.4 if the value passed to the constructor is 0. You can assume that any value we supply through the command line will be > 0.

We highly encourage you to experiment with different values of λ and see how it effects the number of clusters which are produced.

1.5 Number of Iterations

The default number of clustering iterations is 10. An iteration is defined as computing 1 E-step and 1-M step of the algorithm (in that order).

You *must* define a new command line option for this parameter: `clustering_training_iterations`. You can add this option by adding the following code to the `createCommandLineOptions` method of `Classify`.

```
registerOption("clustering_training_iterations", "int", true, "The number of lambda-means EM iterations.");
```

You can then read the value from the command line by adding the following to the main method of `Classify`:

```
int clustering_training_iterations = 10;
if (CommandLineUtilities.hasArg("clustering_training_iterations"))
    clustering_training_iterations = CommandLineUtilities.getOptionValueAsInt("clustering_training_iterations");
```

1.6 Implementation Details

1.6.1 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.

1.6.2 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 $r_{nk} = 0,\ \forall n$. In this case, you should set $\mu_k = \mathbf{0}$. Do not remove empty clusters.

1.6.3 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. One option is to simply use arrays of a fixed size, and expand 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 use a Java data structure such as `ArrayList` which does not have a fixed size.

1.7 Evaluation

For clustering, we cannot simply use accuracy against true labels to evaluate the output, since your prediction outputs are simply the indices of clusters, which could be arbitrary. We will evaluate your output using an information-theoretic metric called *variation of information* (VI), which can be used to measure the dependence of the cluster indices with the true labels. The variation of information between two random variables Y and $\hat{Y}$ is defined as: $H(Y|\hat{Y}) + H(\hat{Y}|Y)$. Lower is better. This will have a value of 0 if there is a one-to-one mapping between the two labelings, which is the best you can do. We have provided a Python script to compute this metric: `cluster_accuracy.py`.

Additionally, we have provided a script that displays the number of unique clusters: `number_clusters.py`.

We will evaluate your implementation on the standard data sets you have been using (`speech`, `bio`, etc.) as well as the multi-class data, `speech.mc`.

2 Analytical (50 points)

1) K-means and Overfitting (11 points) Suppose we apply the K-means algorithm to a data set of n examples, $x_1, ..., x_n$. The K-means algorithm aims to partition the n observations into k sets ($k \leq n$) $S = \{S_1, S_2, , S_k\}$ so as to minimize the within-cluster sum of squares

$$\min_{S=\{S_1,...,S_k\}} \sum_{j=1}^{k} \sum_{x_i \in S_j} \|x_j - \mu_j\|_2^2, \tag{4}$$

where μ_j is the mean of points in S_j.

(a) Let γ_k denote the minimum of (4). 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, $x \to \phi(x)$. Then we can rewrite (4) as

$$\min_{S=\{S_1,...,S_k\}} \sum_{j=1}^{k} \sum_{x_i \in S_j} \|\phi(x_j) - \alpha_j\|_2^2, \tag{5}$$

where α_j is the mean of points in S_j in the feature space, i.e,

$$\alpha_j = \frac{1}{|S_j|} \sum_{x_i \in S_j} \phi(x_i). \tag{6}$$

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

2) Variable Selection for K-means (11 points) Let's consider the same setup as the previous problem. We can modify K-means objective with ℓ_1 regularization. More specifically, we consider

$$\min_{S=\{S_1,\ldots,S_k\}} \sum_{j=1}^{k} \sum_{x_i \in S_j} \|x_j - \mu_j\|_2^2 + \lambda \sum_{g=1}^{k} \|\mu_g\|_1, \tag{7}$$

where $\lambda > 0$ is the regularization parameter.

(a) Please derive the update formula for the M-step and describe the result. You may need the fact that,

$$\hat{v} = \arg\min_{v} \frac{1}{2}\|u - v\|_2^2 + \lambda \|v\|_1 \tag{8}$$

has a closed form solution

$$\hat{v} = [\hat{v}_j]_{j=1}^d = [\text{sign}(u_j) \max\{|u_j| - \lambda, 0\}_{j=1}^d]. \tag{9}$$

(b) The ℓ_1 regularization encourages sparsity of μ_j. If the results have the m-th entries of all μ_j's equal to zero, what will this imply?

3) Variants of K-means (10 points) Another variant is to change the distance metric to be the L1 distance, which is more robust to outliers. In this case, we are minimizing:

$$\min_{S=\{S_1,\ldots,S_k\}} \sum_{j=1}^{k} \sum_{x_i \in S_j} \|x_j - \mu_j\|_1 \tag{10}$$

where μ_j is the center of S_j w.r.t. to L1 distance.

(a) Please derive the update formula for the M-step.

(b) This algorithm is also called K-medians. Why?

4) EM Clustering (10 points) Suppose we have a Naive Bayes clustering algorithm and a K-means clustering algorithm. Consider the following statement:
Each instance is assigned, or has the highest probability of assignment, to the cluster whose mean is closest to the instance.

Is this statement true:

(a) After the E-Step?

(b) After the M-Step?

Answer true or false and why for (a) and (b) for both the Naive Bayes clustering and the K-means clustering algorithms.

5) Online GMM (8 points) We are given a data set $x_1, ..., x_N$ and want to run Gaussian Mixture Model (GMM) clustering with K clusters and I iterations. However, in this case N is very large and we cannot afford to keep all N in memory. Suppose instead that we used a streaming version of a GMM, in which examples are processed one at a time. Optionally, we can make multiple passes over the data set. Suppose that for each example, we compute a single E-step and M-step based on the given example. After these two steps, we discard the example and move on to the next one.

(a) Write the E-step and M-step updates.

(b) If we ran this algorithm many times over the data set (instead of a single pass), would it converge to the same solution as batch GMM?

(c) What is the total memory requirement for streaming GMM (in terms of examples and parameters)?

3 What to Submit

In each assignment you will submit two things.

1. **Code:** Your code as a zip file named `library.zip`. **You must submit source code (.java files)**. We will run your code using the exact command lines described above, so make sure it works ahead of time. Remember to submit all of the source code, including what we have provided to you.

2. **Writeup:** Your writeup as a **PDF file** (compiled from latex) containing answers to the analytical questions asked in the assignment. Make sure to include your name in the writeup PDF and use the provided latex template for your answers.

Make sure you name each of the files exactly as specified (library.zip and writeup.pdf).

To submit your assignment, visit the "Homework" section of the website (http://www.cs475.org/.)

4 Questions?

Remember to submit questions about the assignment to the appropriate group on the class discussion board: http://bb.cs475.org.