

Coursera Beta

- Uday Upreti
- Contact Us
- My Contributions
- Log Out

ML:Clustering

From Coursera

Contents

- 1 Unsupervised Learning: Introduction
- 2 K-Means Algorithm
- 3 Optimization Objective
- 4 Random Initialization
- 5 Choosing the Number of Clusters
- 6 Bonus: Discussion of the drawbacks of K-Means

Unsupervised Learning: Introduction

Unsupervised learning is contrasted from supervised learning because it uses an **unlabeled** training set rather than a labeled one.

In other words, we don't have the vector y of expected results, we only have a dataset of features where we can find structure.

Clustering is good for:

- Market segmentation
- Social network analysis
- Organizing computer clusters
- Astronomical data analysis

K-Means Algorithm

The K-Means Algorithm is the most popular and widely used algorithm for automatically grouping data into coherent subsets.

1. Randomly initialize two points in the dataset called the *cluster centroids*.
2. Cluster assignment: assign all examples into one of two groups based on which cluster centroid the example is closest to.
3. Move centroid: compute the averages for all the points inside each of the two cluster centroid groups, then move the cluster centroid points to those averages.
4. Re-run (2) and (3) until we have found our clusters.

Our main variables are:

K (number of clusters)

Training set $x^{(1)}, x^{(2)}, \dots, x^{(m)}$

Where $x^{(i)} \in \mathbb{R}^n$

Note that we **will not use** the $x_0 = 1$ convention.

The algorithm:

```
Randomly initialize K cluster centroids mu(1), mu(2), ..., mu(K)
Repeat:
  for i = 1 to m:
    c(i) := index (from 1 to K) of cluster centroid closest to x(i)
  for k = 1 to K:
    mu(k) := average (mean) of points assigned to cluster k
```

The **first for-loop** is the 'Cluster Assignment' step. We make a vector c where $c(i)$ represents the centroid assigned to example $x(i)$.

We can write the operation of the Cluster Assignment step more mathematically as follows:

$$c^{(i)} = \operatorname{argmin}_k \|x^{(i)} - \mu_k\|^2$$

That is, each $c^{(i)}$ contains the index of the centroid that has minimal distance to $x^{(i)}$.

By convention, we square the right-hand-side, which makes the function we are trying to minimize more sharply increasing. It is mostly just a convention. But a convention that helps reduce the computation load because the Euclidean distance requires a square root but it is canceled.

Without the square:

$$\|x^{(i)} - \mu_k\| = \left\| \sqrt{(x_1^i - \mu_{1(k)})^2 + (x_2^i - \mu_{2(k)})^2 + (x_3^i - \mu_{3(k)})^2 + \dots} \right\|$$

With the square:

$$\|x^{(i)} - \mu_k\|^2 = \left\| (x_1^i - \mu_{1(k)})^2 + (x_2^i - \mu_{2(k)})^2 + (x_3^i - \mu_{3(k)})^2 + \dots \right\|$$

...so the square convention serves two purposes, minimize more sharply and less computation.

The **second for-loop** is the 'Move Centroid' step where we move each centroid to the average of its group.

More formally, the equation for this loop is as follows:

$$\mu_k = \frac{1}{n} [x^{(k_1)} + x^{(k_2)} + \dots + x^{(k_n)}] \in \mathbb{R}^n$$

Where each of $x^{(k_1)}, x^{(k_2)}, \dots, x^{(k_n)}$ are the training examples assigned to group μ_k .

If you have a cluster centroid with **0 points** assigned to it, you can randomly **re-initialize** that centroid to a new point. You can also simply **eliminate** that cluster group.

After a number of iterations the algorithm will **converge**, where new iterations do not affect the clusters.

Note on non-separated clusters: some datasets have no real inner separation or natural structure. K-means can still evenly segment your data into K subsets, so can still be useful in this case.

Optimization Objective

Recall some of the parameters we used in our algorithm:

$c^{(i)}$ = index of cluster (1,2,...,K) to which example $x^{(i)}$ is currently assigned

μ_k = cluster centroid k ($\mu_k \in \mathbb{R}^n$)

$\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned

Using these variables we can define our **cost function**:

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

Our **optimization objective** is to minimize all our parameters using the above cost function:

$$\min_{c, \mu} J(c, \mu)$$

That is, we are finding all the values in sets c , representing all our clusters, and μ , representing all our centroids, that will minimize **the average of the distances** of every training example to its corresponding cluster centroid.

The above cost function is often called the **distortion** of the training examples.

In the **cluster assignment step**, our goal is to:

Minimize $J(\dots)$ with $c^{(1)}, \dots, c^{(m)}$ (holding μ_1, \dots, μ_K fixed)

In the **move centroid** step, our goal is to:

Minimize $J(\dots)$ with μ_1, \dots, μ_K

With k-means, it is **not possible for the cost function to sometimes increase**. It should always descend.

Random Initialization

There's one particular recommended method for randomly initializing your cluster centroids.

1. Have $K < m$. That is, make sure the number of your clusters is less than the number of your training examples.
2. Randomly pick K training examples. (Not mentioned in the lecture, but also be sure the selected examples are unique).
3. Set μ_1, \dots, μ_k equal to these K examples.

K-means **can get stuck in local optima**. To decrease the chance of this happening, you can run the algorithm on many different random initializations. In cases where $K < 10$ it is strongly recommended to run a loop of random initializations.

```

for i = 1 to 100:
  randomly initialize k-means
  run k-means to get 'c' and 'm'
  compute the cost function (distortion) J(c,m)
pick the clustering that gave us the lowest cost
  
```

Choosing the Number of Clusters

Choosing K can be quite arbitrary and ambiguous.

The elbow method: plot the cost J and the number of clusters K . The cost function should reduce as we increase the number of clusters, and then flatten out. Choose K at the point where the cost function starts to flatten out.

However, fairly often, the curve is **very gradual**, so there's no clear elbow.

Note: J will **always** decrease as K is increased. The one exception is if k-means gets stuck at a bad local optimum.

Another way to choose K is to observe how well k-means performs on a **downstream purpose**. In other words, you choose K that proves to be most useful for some goal you're trying to achieve from using these clusters.

Bonus: Discussion of the drawbacks of K-Means

From StackExchange (<http://stats.stackexchange.com/questions/133656/how-to-understand-the-drawbacks-of-k-means>) This links to a discussion that shows various situations in which K-means gives totally correct but unexpected results.

Next: Dimensionality Reduction Back to Index: Main

Retrieved from "<https://share.coursera.org/wiki/index.php?title=ML:Clustering&oldid=34220>"

Category: ML:Lecture Notes

- This page was last modified on 2 September 2016, at 08:10.
- This page has been accessed 20,327 times.
- [Privacy policy](#)
- [About Coursera](#)
- [Disclaimers](#)