Clustering

Andrés Ponce

May 30, 2020

Here we study *clusters* of data and how we can use them to our advantage.

Supervised vs. Unsupervised Learning

Supervised Learning

Using supervised learning, we always have access to the actual answer which we use to compare. For example, we would have an input vector **x** and a target vector **t**, then we can see how far off a potential guess is from the actual answer. We can see how accurate a model is in predicting the values of a function or how accurately a model classifies data. The important thing is that we also have a target vector.

The goal of supervised learning is to learn a **mapping** between the input vector and the target vector (or as close to target vector as possible.)

Unsupervised Learning

For unsupervised learning, we only have access to **x**, the input data vector. The goal of unsupervised learning is to then find a **relation** or otherwise hidden structure in the data. Some common use cases for unspupervised learning include: clustering, dim. reduction, density estimation, data generation, etc...

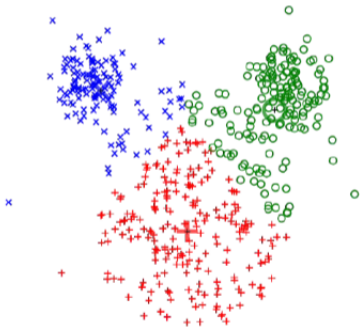

Figure 1: We split the data set into k clusters. Data points in each cluster are more similar to each other than to points in other clusters.

Unsupervised Learning applications

Clustering

Goal: given a data set, separate the data into **clusters**, or regions where many data points gather. The points in these areas are more similar to themselves than they are to members of other clusters.

Dimensionality Reduction

Similar to the second homework assignment, we take a data set with k dimensions of data and try to project it onto a lower data space. This reduces the complexity of solving problems on this data.

Density Estimation

Given a set of data points, try to find a probability density function.

Data reconstruction

Build models that can efficiently **encode** and **decode** the data once its sent. The encoder decreases the dimensionality of the data, whereas the decoder will increase the dimensionality to reconstruct the original data.

Data generation

Here we can generate new images from a high dimensional distribution. [1]

[1] Is this how the NVIDIA program generated new faces?

Clustering

As mentioned, these types of problems ask us to find related groups of data points in D dimensional data. We introduce the dimensional vectors μ_k for each of the k clusters. These vectors are directed at the center of the data points in the cluster. The objective is to find the configuration such that the sum of squares distance between each data point and its vector is at a minimum. We also have an indicator variables $r_{nk} \in \{0, 1\}$ if $\mathbf{x_n}$ is assigned to the k-th cluster.

For clustering, the objective function is called a **distortion measure**, and is represented by [2]

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{x_n} - \mu_k\|^2$$

[2] This equation basically sums the squared distance between a data point if it is assigned to cluster k and all the dimensional vectors. I guess we try to find the configuration such that this is minimized?

The main steps in the k-clustering algorithm can be summarized as follows:

1. Choose some initial values for clusters μ_k
2. Keep the clusters $\{\mu_k\}_{k=1}^{k}$ fixed and try to minimize J w.r.t. assignment $\{\mu_k\}_{k=1}^{K}$
3. Keep assignment $\{\mu_k\}_{k=1,n=1}^{K,N}$ fixed and minimize J w.r.t. clusters $\{\mu_k\}$
4. Repeat steps 2 and 3 until convergence.

This algorithm is really more reflective of a broader type of algorithms: **Expectation-Maximization** algorithms. These algorithms consists of two steps: We then continuously keep modifying our prediction for μ_k for each k, either by using the entire set of points for a

cluster or only a subset. Similar to what we did in homework 1, we can adjust the new dimensional vector with the formula:

$$u_k^{new} = u_k^{old} + \eta_n(x_n - \mu_k^{old})$$

k-medoids clustering

This clustering method is the same as the regular *k*-means except that:

1. The objective function is now

$$\bar{J} = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk} V(x_n, \mu_k)$$

   This new objective function allows us to generalize so we are not required to use Euclidean distance as a measure of dissimlarity. Using a straightforward Euclidean distance could make our estimates susceptible to outliers.

2. Now in the M-step, μ_k has to be a data point. We set μ_k eqaul to the data point with the shortest average distance to all other points.

Notice that *k*-medoids could be quite expensive to run, since we have to evaluate V every iteration for every data point.

Image segmentation using k-means clustering

The objective of *k*-means clustering on images is to separate them into regions whose pixels have similar appearance. One big use case for image segmentation would be compression. However, there are some diffrences from what we have mentioned: since we are working with images, our input vector $\mathbf{x_n}$ is comprised of values $\in \{0, 1\}$, so we are not finding the Euclidean distance necessarily. We could apply this to do **lossless** compression or **lossy** compression, where we do not allow any missing informationn for the original image values and where we allow an acceptable amount, respectively. With lossy compression, we could set each individual image value to the value of its nearest μ_k.

Gaussian Distribution

To remind, a probability distribution is **Gaussian** if values vollow the formula [3]

$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\}$$

[3] Here μ is the mean vector and Σ is the covariance matrix

Although Gaussian distributions appear quite frequently in nature (e.g. heights and weights of populations) or in more human situations (e.g. testing scores), some patterns of data are not fit to be described by a Gaussian distribution.

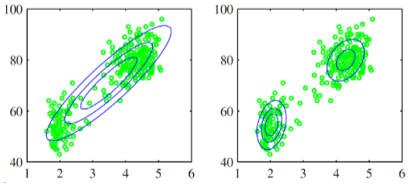

Figure 2: These distributions do not reflect the standard "bell curve" normally associated with Gaussian distributions.

Gaussian Mixture Model

In the context of clustering, for k clusters we could have up to k different Gaussian distributions controlling the data points that belong to that cluster.

We introduce a variable z such that for each data point $\mathbf{x}_n$, there is also a z_n. Also $z \in \{0, 1\}$ and $\sum_k z_k = 1$. [4]

Thus to calculate the probability **x** belongs to a certain cluster, we have to calculate the joint and marginal probability of x and z.

The probability **z** is equal to 1 depends on the **mixing coefficient** of the distributions, or how separated they are. The marginal probability of **z** is expressed by

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$$

The conditional probability $p(x|\mathbf{z})$ would then be the product of $\mathcal{N}(x|\mu_k, \Sigma_k)$.

We have already modeled the conditional probability that $mathbf z = 1$ given **x**, and now we model the other conditional probability: $p(\mathbf{z}|x)$. This term γ we call the **responsibility** that k takes for explaining x. Although the formula is complicated, it is not hard: [5]

$$\gamma(z_k) = \frac{p(z_{k=1})p(x|z_k = 1)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x|\mu_j, \Sigma_j)}$$

[4] This means that for every x_n, it will have a value such that z indicates which group it belongs to. From the classification section, this is a **1-of-k** representation.

[5] What this formula essentially says is that the probability that z is correct given x is equal to the probability x belongs to cluster k over the probability that it belongs to any other cluster.