Unsupervised Learning

MInDS @ Mines

Unsupervised learning is the type of machine learning where we do not use labeled data or necessarily have knowledge of a ground truth. This makes evaluating our models tricky since we can never be certain of what the correct answer really is. There are a variety of applications of unsupervised learning and in this lecture we will discuss clustering. We will also go over a simple clustering method called k-Means clustering.

Overview

Unsupervised learning is a subset of machine learning where the model is not provided with ground truth labels for the problem. The model is expected to determine some patterns in the data without any guidance or "supervision". There are two main problem types where we can use unsupervised learning methods; one begins with building a model for data that we have ground truth labels for, and the other is building a model without knowing the ground truth result of our data. The first type, withholding ground truth, allows us to apply unsupervised learning methods to a supervised learning problem which may give us a new understanding of our data based on the patterns the model recognizes. It also allows us to evaluate or better understand the unsupervised learning methods we use. The second type, unknown ground truth, is a more true version of the problems unsupervised learning methods are developed to solve. With this approach, we couldn't apply supervised learning methods to the same problem.

One application of unsupervised learning is clustering. Clustering is the grouping of similar data samples together to define representative groups, called clusters. There are many approaches to clustering and they generally differ based on the grouping approach, or how we determine samples' similarity, and how we determine the number of clusters.

Clustering problems are analogous to classification problems. In classification problems, the model determines the class a sample belongs to while in clustering problems, the model determines the cluster a sample belongs to.

Evaluation Metrics

Depending on the problem type, we can evaluate our model using a variety of metrics. When we are withholding ground truth from the model, our evaluation metrics may be more representative since we actually know the correct answer to the problem. However, when we do not know the ground truth, our evaluation metrics are somewhat fuzzy. They determine how closely our result follows some expected pattern or examine the variability in our data. Both these evaluation techniques are useful but may be very different from ground truth evaluation.

Withholding ground truth

Withholding ground truth from the model means that the model can not use the correct labels of our data to determine its parameters. The model must recognize a pattern in the data and then we can utilize the available ground truth metric to evaluate its performance. There are several metrics to achieve this, namely:

- Adjusted rand index
- Adjusted mutual information score
- Homogeneity, completeness, and V-measure

We define a problem of n samples having s clusters with r pre-determined classes. C is the grouping of samples based on their classes and P is the grouping based on their predicted clusters. We can develop a contingency table / matrix to display the number of samples from each class in each predicted cluster.

A contingency matrix in clustering is slightly analogous to a confusion matrix for classification.

Adjusted rand index. The rand index is a measure of consistency across element groupings. It examines each pair of points from the samples and tracks how many pairs are within the same grouping in both C and P, and how many are within different groupings in both C and P. It is calculated as,

$$rand = \frac{ss + dd}{ss + dd + sd + ds} = \frac{ss + dd}{\binom{n}{2}}, \tag{1}$$

where ss represents the number of sample pairs that are in the same cluster in C and in P, dd represents the number of sample pairs that are in different clusters in C and in P, sd represents the number of sample pairs that are in the same cluster in C but different in P, and ds represents the number of sample pairs that in different clusters in C but the same cluster in P.

We can correct for chance when using the rand index by using the adjusted rand index. The adjusted rand index can be calculated as,

$$\text{ARI} = \frac{\sum_{ij}\binom{n_{ij}}{2} + \left[\sum_i \binom{w_i}{2}\sum_j \binom{x_j}{2}\right]/\binom{n}{2}}{\left[\sum_i \binom{w_i}{2} + \sum_j \binom{x_j}{2}\right]/2 - \left[\sum_i \binom{w_i}{2}\sum_j \binom{x_j}{2}\right]/\binom{n}{2}} \tag{2}$$

The following is an example of a contingency matrix with C and P as the two groupings (ground truth and predicted clusters).

	c_1	c_2	$\dots$	c_s	Sums
p_1	n_{11}	n_{12}	$\cdots$	n_{1s}	w_1
p_2	n_{21}	n_{22}	$\cdots$	n_{2s}	w_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
p_r	n_{r1}	n_{r2}	$\cdots$	n_{rs}	w_r
Sums	x_1	x_2	$\dots$	x_s	

Adjusted mutual information score. For mutual information, we begin by defining $P_U(i)$ as the probability that a randomly selected object belongs to class U_i of set U and $P_{U,V}(i,j)$ as the probability that a randomly selected object belongs to class U_i of set U and class V_j of set V. Entropy of a set U is defined as,

$$H(U) = -\sum_{i=1}^{|U|} P_U(i)\log(P_U(i)). \tag{3}$$

The mutual information between two sets U and V is then defined as,

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P_{U,V}(i,j)\log(\frac{P_{U,V}(i,j)}{P_U(i)P_V(j)}). \tag{4}$$

We can normalize the mutual information using each set's entropy to get values from 0 to 1 as,

$$NMI(U,V) = \frac{MI(U,V)}{\sqrt{H(U)H(V)}}. \tag{5}$$

To adjust for the chance of each class, we can use the same approach we used for the rand index. We won't go into the details of how this is calculated but the general idea is,

$$\text{AMI} = \frac{\text{MI} - \text{Expected MI}}{\text{Max Entropy of U or V} - \text{Expected MI}}. \tag{6}$$

Homogeneity, completeness, and V-measure. Homogeneity measures how many values in a particular cluster are actually from one class whereas completeness measures how many values in a particular class are predicted in the same cluster. The V-measure is the hyperbolic mean of the two values. To calculate homogeneity and completeness we expand on our entropy definition,

Homogeneity, completeness, and the V-measure are analogous to precision, recall, and the F1-score for supervised learning.

$$H(C) = -\sum_{i=1}^{|C|}\frac{n_i}{n}\log(\frac{n_i}{n}). \tag{7}$$

Note that $\frac{n_i}{n}$ is equivalent to the probability of class i in the set, similar to our previous definition. We also expand our definition to conditional entropy,

$$H(C \mid P) = -\sum_{i=1}^{|C|}\sum_{j=1}^{|P|}\frac{n_{i,j}}{n}\log(\frac{n_{i,j}}{n_j}). \tag{8}$$

We can now define homogeneity, completeness and the V-measure as,

Homogeneity and completeness are each not symettric. However, the homogeneity of two clusterings, $h(A,B)$ of a dataset is the completeness of the two clusterings in reverse order, $c(B,A)$.

$$\begin{aligned}
\text{Homogeneity} &= 1 - \frac{H(C \mid P)}{H(C)} \\
\text{Completeness} &= 1 - \frac{H(P \mid C)}{H(P)} \\
\text{V-Measure} &= \frac{2 \times \text{Homogeneity} \times \text{Completeness}}{\text{Homogeneity} + \text{Completeness}}
\end{aligned} \tag{9}$$

Unknown ground truth

When we do not know the ground truth values for our data, it is difficult to assess the performance of our model. There are fewer metrics for evaluating models with unknown ground truth. We will only cover using the silhouette coefficient for scoring to introduce the approach of these metrics.

Silhouette coefficient. The silhouette coefficient of a sample provides a ratio between the distance to the mean of the selected cluster and the next

closest cluster. This helps us estimate how close a point is to the next nearest cluster. First we define $\hat{w}_{i,k}$ as the distance between a sample i belonging to class k and the mean of class k,

$$\hat{w}_{i,k} = ||x_i - \hat{x}_k||_p^r, \tag{10}$$

where $\hat{x}_k$ is the mean of all the points in cluster k, and p and r are determined based on the distance metric we wish to utilize.

The silhoutte score is customizable in that we can select which distance metric to use. Intuitively, the silhouette score is the difference between the distance from the point to its labeled cluster and the next closest cluster divided by the maximum of the two.

We also define $\hat{d}_{i,k}$ as the distance from a point i to mean of the next closest cluster to k,

$$\hat{d}_{i,k} = ||x_i - \hat{x}_l||_p^r, \tag{11}$$

where $\hat{x}_l$ is the mean of all the points in cluster l, the closest cluster to k. The silhouette coefficient for a single sample i belonging to cluster k is,

$$\text{Silhouette}_i = \frac{\hat{d}_{i,k} - \hat{w}_{i,k}}{\max(\hat{w}_{i,k}, \hat{d}_{i,k})} \tag{12}$$

The silhouette score of clustering output is the mean of all the silhouette coefficients of the samples in the data.

Clustering

Clustering is about determining a pattern in our data and understanding if many data samples share similar properties. By grouping our data into clusters, we can develop an understanding of how our data exists in nature and begin to model that effectively.

k-Means Clustering

One of the simplest clustering methods is the k-Means clustering method. k-Means has a hyperparameter k which is the number of clusters to group the data into. k-Means determines cluster centroids, μ that solve the objective,

$$\min_{\mu} \sum_{i=1}^{k} \sum_{x_j \in C_i} ||x_j - \mu_i||_2^2, \tag{13}$$

where μ_i is the centroid of cluster i and C_i is the subset of points that belong to cluster i. k-Means minimizes the Euclidean distance between each point and the centroid of its labeled cluster. It does so using the following algorithm,

Note that the learned centroids for the clusters are not necessarily actual points from the data but belong to the same space.

Algorithm 1: K Means Solution Algorithm

Input: Features from data

Output: Centroids of the k clusters

1 Initialize cluster centroids (randomly or using a particular strategy)
2 **while** *cluster centroids update not converge* **do**
3 | Determine cluster for each point based on distance to centroids
4 | Update centroids' location as the mean of points in the cluster
5 **end**