Week 8: Clustering

Instructor: Sergey Levine

1 Unsupervised Learning

So far, we've discussed a variety of supervised learning algorithms: algorithms that predict a label y given attributes $\mathbf{x}$. In this next unit, we will study unsupervised learning methods. In contrast to supervised learning, we won't have labels y, only inputs $\mathbf{x}$. Our goal then is not to predict a label, but rather to recover the structure in the data. This structure can then be used for a variety of subsequent applications: we can use it to try to understand something about the phenomenon that gives rise to the data, visualize the data to make it easier to interpret, or use the learned structure as the basis for subsequent (supervised) learning methods. This latter technique is sometimes referred to as semi-supervised learning. We'll see some specific examples of applications in later lectures.

2 Clustering

The first class of unsupervised learning methods we'll study will be clustering methods. In clustering, our goal is to recover a very particular structure. Our data will consist of real-valued vectors $\mathbf{x} \in \mathbb{R}^D$, and our goal will be to find a small number of clusters that describe the data distribution: that is, group the data into a small set of "groups." Some example applications in practice might include clustering customers of an online service based on their usage habits to identify particular categories or types of customers, or learning the structure of web traffic in order to be able to detect anomalous and potentially malicious activity (we'll talk about this later when we discuss probabilistic clustering).

In order to group our data into a discrete set of groups (clusters), we first need some definition of what constitutes a "good" group: just like with all other machine learning problems, we need an objective. If we assume that we have K clusters $C_1, \ldots, C_K$, one simple objective might be to minimize the sum of average intragroup distances per cluster – that is, entries that are assigned to the same group should all be close to one another:

$$\mathcal{L}(C_1, \ldots, C_K) = \sum_{k=1}^{K} \frac{\sum_{\mathbf{x}_i \in C_k, \mathbf{x}_j \in C_k} \frac{1}{2} \|\mathbf{x}_i - \mathbf{x}_j\|^2}{\|\{i : \mathbf{x}_i \in C_k\}\|}.$$

Question. Recall that we need to define the data, the hypothesis space, the objective, and the algorithm. What is the hypothesis space for this learning problem?

Answer. Since we are trying to find clusters $C_1, \ldots, C_K$, one simple way to parameterize the hypothesize space is to associate a discrete cluster label y_i with each datapoint $\mathbf{x}_i$. Here, y_i is categorical in the set $\{1, \ldots, K\}$, and we have $\mathbf{x}_i \in C_k$ if and only if $y_i = k$. The choice of y here is intentional – we'll see the connection to labels later.

Now all we have to do is design an algorithm that will minimize the distances objective with respect to the cluster assignments. This is a bit tricky, because the total number of possible cluster assignments is K^N, since each point might belong to a different cluster, so we can't possibly evaluate all possible cluster assignments. In fact, it can be shown that optimal assignment is NP-hard. However, we can devise an algorithm that is at least locally optimal (just like we did with neural networks), meaning that it converges to a solution where reassigning any of the points to a different cluster will make the objective worse. Unfortunately, we can't quite use gradient ascent or descent in this case, since we're optimizing over discrete variables.

Here is a simple idea for an algorithm: repeatedly search through all the points and, if there is some cluster that is *closer* to that point than the one it is currently in, we'll simply reassign it to that cluster. Once there is no point for which reassigning it to a closer cluster is possible, we know that we have reached a local optimum.

Now, doing this exactly is a bit costly, since we need to step through all N points each time we want to find a new cluster for each point, so just one iteration of this simple algorithm is $O(N^2)$. But we can do something much less costly: for each cluster, we'll define a centroid $\mathbf{c}_k$, and then instead of exhaustively evaluating all distances to all other points for each of the N points, we'll only check its distance to the centroids, and assign it to the nearest centroid. Since we have K centroids and N points, each iteration requires only $O(NK)$ time. The algorithm then looks like this:

Algorithm 1 K-means clustering

1: Initialize cluster assignments y_i with random integers in $\{1, \ldots, K\}$
2: **while** not converged **do**
3: $\quad \mathbf{c}_k \leftarrow \frac{1}{\|i:y_i=k\|} \sum_{i:y_i=k} \mathbf{x}_i$ (average all points with $y_i = k$)
4: $\quad y_i \leftarrow \arg\min_k \|\mathbf{x}_i - \mathbf{c}_k\|^2$ (assign each point to nearest cluster)
5: **end while**

This algorithm is very simple, and in fact we can prove that it optimizes the intraclass distances objective. To prove this, we'll first consider a surrogate

objective that includes the centroids $\mathbf{c}_k$:

$$\hat{\mathcal{L}}(y_1, \ldots, y_N, \mathbf{c}_1, \ldots, \mathbf{c}_K) = \sum_{k=1}^{K} \sum_{i:y_i=k} \|\mathbf{x}_i - \mathbf{c}_k\|^2$$

We'll start by showing that K-means locally minimizes this objective. K-means consists of just two steps, and the way we show that it minimizes this objective is by proving that each individual step decreases the objective. The first step clearly decreases the objective, because if we hold the cluster assignments fixed and optimize with respect to $\mathbf{c}_k$, we can compute the gradient and solve:

$$\frac{d\hat{\mathcal{L}}}{d\mathbf{c}_k} = -\sum_{i:y_i=k} 2(\mathbf{x}_i - \mathbf{c}_k) = 0 \Rightarrow \sum_{i:y_i=k} \mathbf{c}_k = \sum_{i:y_i=k} \mathbf{x}_i \Rightarrow \mathbf{c}_k = \frac{\sum_{i:y_i=k} \mathbf{x}_i}{\|i : y_i = k\|}$$

The second step decreases the objective because we explicitly choose cluster assignments that will minimize the distance between each $\mathbf{x}_i$ and the chosen cluster center $\mathbf{c}_k$. So if each step of K-means can only decrease the objective, then the algorithm will only terminate (converge) once the objective is at a local minimum.

To show that K-means in fact also minimizes the sum of intracluster distances, we can show that the sum of intracluster distances is in fact equal to the surrogate objective used by K-means:

$$\begin{aligned}
\sum_{k=1}^{K} \frac{\sum_{\mathbf{x}_i \in C_k, \mathbf{x}_j \in C_k} \frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\|\{i : y_i = k\}\|} &= \sum_{k=1}^{K} \frac{\sum_{\mathbf{x}_i \in C_k, \mathbf{x}_j \in C_k} \frac{1}{2}\|\mathbf{x}_i - \mathbf{c}_k + \mathbf{c}_k - \mathbf{x}_j\|^2}{\|\{i : y_i = k\}\|} \\
&= \sum_{k=1}^{K} \frac{\sum_{\mathbf{x}_i \in C_k, \mathbf{x}_j \in C_k} \frac{1}{2}\left[\|\mathbf{x}_i - \mathbf{c}_k\|^2 - 2(\mathbf{x}_i - \mathbf{c}_k) \cdot (\mathbf{x}_j - \mathbf{c}_k) + \|\mathbf{c}_k - \mathbf{x}_j\|^2\right]}{\|\{i : y_i = k\}\|} \\
&= \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in C_k} \left[\|\mathbf{x}_i - \mathbf{c}_k\|^2 - \frac{(\mathbf{x}_i - \mathbf{c}_k) \cdot \sum_{\mathbf{x}_j \in C_k} (\mathbf{x}_j - \mathbf{c}_k)}{\|\{i : y_i = k\}\|}\right].
\end{aligned}$$

However, since $\mathbf{c}_k$ is the average of the points $\mathbf{x}_j$ in the cluster k, we must have $\sum_{\mathbf{x}_j \in C_k} (\mathbf{x}_j - \mathbf{c}_k) = 0$, so the last term is equal to zero, and therefore we have

$$\sum_{k=1}^{K} \frac{\sum_{\mathbf{x}_i \in C_k, \mathbf{x}_j \in C_k} \frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\|\{i : y_i = k\}\|} = \sum_{k=1}^{K} \sum_{i:y_i=k} \|\mathbf{x}_i - \mathbf{c}_k\|^2.$$

Therefore, we know that if we minimize the distance to the centroids, we'll also minimize the intraclass distance averaged over all of the clusters.