CSE 347 Lecture 26 (v2)

Smoothed Analysis

1. k-means clustering & Lloyd's Algorithm
2. Smoothed Analysis
3. Smoothed running time of Lloyd's Algorithm (in high dimension)

"Clustering" is a widely used "unsupervised" machine learning problem formulation. Intuitively, we wish to discover some structure in a data set by identifying clumps of points that are near each other, but perhaps far from the points in another clump. These sets of points are called "clusters," and the hope is usually that the membership in a cluster indicates some interesting property of its constituent points that was not explicitly represented in the data set. For example, the points may correspond to documents, represented by a list of which words do or do not appear in that document. One may hope that the clusters represent the topics of the documents, for example. One can hope for many such things, and in general these hopes may not come true. But, clustering is widely used. To be more precise, here is one widely-used objective for clustering, known as <u>k-means</u>: for a set of points $\mathcal{P} \subseteq \mathbb{R}^n$, we wish to find a collection

	Words			
	Word 1	Word 2	⋯	Word n
Doc. 1	0	1		0
Doc. 2	1	1		0
⋮				
Doc m	1	0		1

of k <u>cluster centers</u> $c_1, \dots, c_k$ such that if $\mathcal{P}_i \subseteq \mathcal{P}$ is the set of points assigned to the center c_i, we minimize $\sum_{i=1}^{k} \sum_{x \in \mathcal{P}_i} \|x - c_i\|^2$ where $\|x - c_i\|$ is the standard, Euclidean distance

i.e., $\|x - c_i\|^2 = \sum_{j=1}^{n} (x_j - c_{ij})^2$

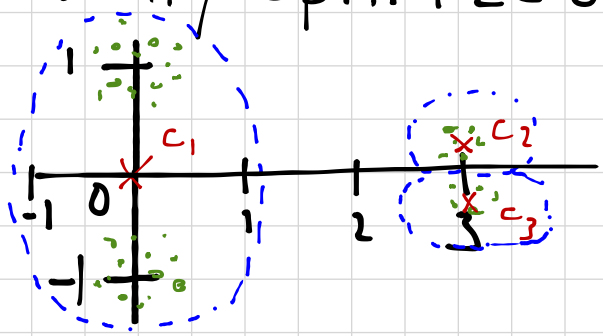
A very widely used algorithm for the k-means objective is Lloyd's Algorithm (so widely used that it is often called "<u>the</u> k-means algorithm," although this is an abuse of the term k-means...)

Given any initial set of k centers $c_1, \dots, c_k \in \mathbb{R}^n$,

Repeat:
- For each $x \in \mathcal{P}$, put x in $\mathcal{P}_i$ s.t. $\|x - c_i\|$ is minimized.
- For $i = 1, \dots, k$, put $c_i = \frac{1}{|\mathcal{P}_i|} \sum_{x \in \mathcal{P}_i} x$

Until no $\mathcal{P}_i$ changes during some iteration.

The first thing to note about k-means is that it does <u>not</u> actually optimize our objective function. Consider:



... where we'd prefer, say, $c_1 = (0, 1)$, $c_2 = (0, -1)$, and $c_3 = (3, 0)$. But, Lloyd's Algorithm may return the shown clustering.

We can at least show that the algorithm does find some "local" optimum, i.e., that the loop terminates.

<u>Lemma 1</u> For a set of points $\mathcal{S}$, let $c(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{x \in \mathcal{S}} x$ be the <u>centroid</u>. Then for any c', $\sum_{x \in \mathcal{S}} \|x - c'\|^2 = \sum_{x \in \mathcal{S}} \|x - c(\mathcal{S})\|^2 + |\mathcal{S}| \cdot \|c(\mathcal{S}) - c'\|^2$

<u>Proof</u>: Write $\sum_{x \in \mathcal{S}} \|x - c'\|^2 = \sum_{x \in \mathcal{S}} \|x - c(\mathcal{S}) + c(\mathcal{S}) - c'\|^2$

$= \sum_{x \in \mathcal{S}} \left(\|x - c(\mathcal{S})\|^2 + 2\,(c(\mathcal{S}) - c')(x - c(\mathcal{S})) + \|c(\mathcal{S}) - c'\|^2 \right)$

$= \sum_{x \in \mathcal{S}} \|x - c(\mathcal{S})\|^2 + 2\,(c(\mathcal{S}) - c') \sum_{x \in \mathcal{S}} (x - c(\mathcal{S})) + |\mathcal{S}| \cdot \|c(\mathcal{S}) - c'\|^2$

since $\sum_{v \in \mathcal{S}} x = |\mathcal{S}| \cdot c(\mathcal{S})$. ▨

Now, why does Lemma 1 imply the loop terminates? If we consider the potential function $\varphi = \sum_{i=1}^{k} \sum_{x \in P_i} \|x - c_i\|^2$, we first observe that it is never negative, and the first step of Lloyd's algorithm re-assigns each x to minimize φ given $c_1, \ldots, c_k$, so it does not increase φ. Likewise, by our Lemma, re-assigning c_i to the centroid of P_i also decreases φ. Thus, φ decreases on every iteration. Since there are at most $k^{|P|}$ possible assignments and they can't repeat, the algorithm terminates.

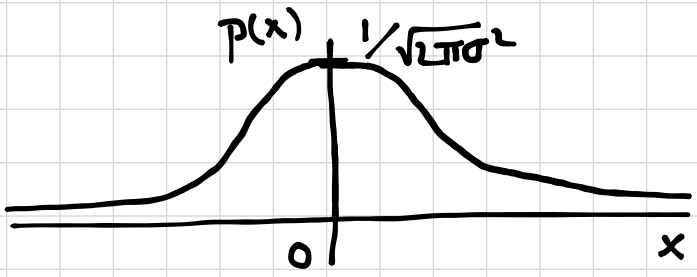
The key question is, how many iterations does this actually take? k-means is popular because empirically, it is very fast. It had been used for nearly 50 years before Arthur and Vassilivitskii found an example set of points (of size m) for which the loop runs for $2^{\Omega(\sqrt{m})}$ iterations.

Smoothed Analysis

How can we reconcile the apparent speed of Lloyd's Algorithm with the fact that it runs in exponential time? An answer is given by "smoothed analysis" — roughly, we assume that the data is a little noisy:

Definition For an algorithm that takes as input an $n \times m$ matrix X, and runs in time $T(X)$, and a noise parameter $\sigma > 0$, the **smoothed running time** on $n \times m$ inputs is $\max_X \mathbb{E}_G[T(X+G)]$ where G is an $n \times m$ matrix of Gaussian (Normal) random variables with mean 0 **and** variance σ^2. (We also consider the distribution of $T(X+G)$.)

Recall that the Gaussian distribution has probability density function $p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{|x-\mu|^2}{2\sigma^2}}$ (here, $\mu = 0$). It is the familiar "bell curve"



Natural variation, measurement error, and so on all empirically seem to roughly follow a Gaussian distribution (The "Central Limit Theorem" in Statistics gives some theoretical justification for this observation.) So, in contrast to average-case analysis, which requires us to specify the distribution of X, it seems reasonable in many cases to assume that the data indeed includes some noise, for a small but non-negligible σ.

How does the noise help us? Precisely, it ensures that the points cannot be carefully arranged:

Lemma 2 If g is a vector of n independent Gaussians of variance σ^2 (and any mean) then for any $y \in \mathbb{R}^n$, $\Pr_g[\|g-y\| \le \varepsilon] < \left(\frac{\varepsilon}{\sigma}\right)^n$.

Proof: Note that at any $x \in \mathbb{R}^n$, the probability density of g at x is at most $\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n$. Moreover, every x with $\|x-y\| \le \varepsilon$ is within ε in each coordinate. So we can crudely approximate the probability:

$$\int_{\|x-y\|\le\varepsilon} \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n e^{-\frac{\|x-\mu\|^2}{2\sigma^2}} dx \le \int_{y_1-\varepsilon}^{y_1+\varepsilon} \cdots \int_{y_n-\varepsilon}^{y_n+\varepsilon} \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n dx_n \cdots dx_1 = \left(\frac{2\varepsilon}{\sqrt{2\pi\sigma^2}}\right)^n < \left(\frac{\varepsilon}{\sigma}\right)^n \blacksquare$$

So, how does this help? It will give us the following

Lemma 3 Every pair of distinct sets $\mathcal{S}, \mathcal{T} \subseteq \mathcal{P}$ satisfies $\|c(\mathcal{S}) - c(\mathcal{T})\| \ge \frac{\varepsilon}{\max\{|\mathcal{S}|,|\mathcal{T}|\}}$ with probability $\ge 1 - 2^{2|\mathcal{P}|}\left(\frac{\varepsilon}{\sigma}\right)^n$ if the points in $\mathcal{P}$ are n-dimensional Gaussians with variance σ^2.

Proof: Fix any pair of distinct sets $\mathcal{S} \neq \mathcal{T}$. Either $\mathcal{S} - \mathcal{T}$ or $\mathcal{T} - \mathcal{S}$ is nonempty; WLOG, let $x^* \in \mathcal{S} - \mathcal{T}$, and consider any outcome for $\mathcal{S} - \{x^*\}$ and $\mathcal{T}$. Then $\Pr\left[\|c(\mathcal{S}) - c(\mathcal{T})\| < \frac{\varepsilon}{|\mathcal{S}|}\right] = \Pr\Big[\|x^* + \underbrace{\sum_{\substack{x\in\mathcal{S},\\ x\neq x^*}} x}_{Y} - \frac{|\mathcal{S}|}{|\mathcal{T}|}\sum_{x\in\mathcal{T}} x\| < \varepsilon\Big]$ which is less than $\left(\frac{\varepsilon}{\sigma}\right)^n$ by Lemma 2.

Now, we observe that there are at most $2^{|\mathcal{P}|}$ choices for each of $\mathcal{S}$ and $\mathcal{T}$, so by a union bound over all pairs, we get $\Pr[\text{any } (\mathcal{S},\mathcal{T}) \text{ has } \|c(\mathcal{S}) - c(\mathcal{T})\| < \frac{\varepsilon}{\max(|\mathcal{S}|,|\mathcal{T}|)}] < 2^{2|\mathcal{P}|}\left(\frac{\varepsilon}{\sigma}\right)^n$ ☒

So, if the dimension n is sufficiently large relative to the number of points, no two centers can be too close: it then follows from Lemma 1 that the number of iterations is bounded:

Theorem If $n \geq \Omega\left(\frac{|\mathcal{P}|}{\log|\mathcal{P}|}\right)$ and $\mathcal{P}$ consists of n-dimensional Gaussian points with variance σ^2 and diameter D, then Lloyd's algorithm terminates in time polynomial in $|\mathcal{P}|$, $\frac{D}{\sigma}$, and $\frac{1}{\delta^{1/n}}$ with probability $1-\delta$ for any $\delta > 0$.

Proof? First note that our potential function $\varphi = \sum_{i=1}^{k} \sum_{x \in \mathcal{P}_i} \|x - c_i\|^2$ is at most $|\mathcal{P}| D^2$ since each $\|x - c_i\| = \|x - \sum_{y \in \mathcal{P}_i} \frac{y}{|\mathcal{P}_i|}\| \leq \frac{1}{|\mathcal{P}_i|} \sum_{y \in \mathcal{P}_i} \|x - y\| \leq D$. Lemma 3 gives that with probability $1 - 2^{2|\mathcal{P}|}\left(\frac{\varepsilon}{\sigma}\right)^n$, distinct cluster centers are at least $\frac{\varepsilon}{|\mathcal{P}|}$ apart, and Lemma 1 shows that φ decreases by at least $\left(\frac{\varepsilon}{|\mathcal{P}|}\right)^2$ on each iteration. Therefore, the algorithm terminates in at most $|\mathcal{P}|^3 \frac{1}{\varepsilon^2} D^2$ iterations with probability $1 - 2^{2|\mathcal{P}|}\left(\frac{\varepsilon}{\sigma}\right)^n$. Now, since $n \geq C \frac{|\mathcal{P}|}{\log|\mathcal{P}|}$ for some $C > 0$, if we take $\varepsilon = \sigma \cdot \left(\frac{\delta}{2^{2|\mathcal{P}|}}\right)^{1/n}$, we find that with probability $1-\delta$, the algorithm terminates in $|\mathcal{P}|^3 \left(\frac{2^{2|\mathcal{P}|}}{\delta}\right)^{\frac{2}{n}} \left(\frac{D}{\sigma}\right)^2$ iterations, which is $\leq |\mathcal{P}|^3 \frac{1}{\delta^{2/n}} 2^{\frac{2 \cdot 2|\mathcal{P}|}{C} \frac{\log|\mathcal{P}|}{|\mathcal{P}|}} \left(\frac{D}{\sigma}\right)^2 = \frac{1}{\delta^{2/n}} |\mathcal{P}|^{3+\frac{4}{C}} \left(\frac{D}{\sigma}\right)^2$, which is indeed polynomial in $\frac{1}{\delta^{1/n}}$, $\frac{D}{\sigma}$, and $|\mathcal{P}|$. ☒

The assumption that the dimension is large makes this analysis easy, but it is not necessary. A more detailed analy-

sis uses Lemma 2 to establish further kinds of configurations are unlikely, that lead to a polynomial bound on the number of iterations in terms of $|P|, k, n, \frac{D}{\sigma}$, and $\frac{1}{\delta}$ for every $|P|, k$, and n. Thus, the presence of some non-negligible noise or variation in the data is sufficient to ensure that the exponential worst-case running time is never observed in practice.

Smoothed analysis has also been used to explain the observed speed of other widely-used algorithms with exponential worst-case running times. Notably, Danzig's Simplex Algorithm for solving Linear Programs was shown to have polynomial running time on inputs with Gaussian noise. This helps explain how it was that the algorithm, which had been the method of choice to solve linear programs in practice since the 1940s, was not noticed to have exponential worst-case running time until the 1970s.