**CPS296.2 Geometric Optimization** **March 1, 2007**

# Lecture 15: Spectral clustering, projective clustering

*Lecturer: Pankaj K. Agarwal* *Scribe: Sam Slee*

In the past two lectures we have been covering the topic of clustering. This lecture continues that discussion with a focus on $k$-means, $k$-median and some related coreset and linear programming results. We first begin by reviewing problem definitions and simple algorithmic approaches to $k$-means.

## 15.1 $k$-means and $k$-median Clustering

Consider a situation where we have $n$ point locations and we wish to place $k$ facilities among these points to provide some service. It is desirable to have these facilities close to the points they are serving, but the notion of "close" can have different interpretations. The **$k$-means problem** seeks to place $k$ facilities so as to minimize the variance between points and the facilities they are assigned to. The **$k$-median problem** is similar, but seeks to minimize the average distance of facility locations to points.

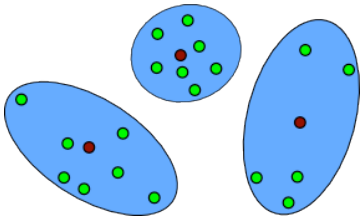

Figure 15.1: $k$=3 clusters with red points chosen as facilities.

### 15.1.1 The $k$-means Problem

Let $S = \{p_1, \ldots, p_n\} \subseteq \Re^d$ be our set of $n$ points. Set $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ as the $k$ locations chosen for facilities. These locations can be at any point in $\Re^d$ (so, we do not have to choose among the $n$ input points for facility locations). Another version of the $k$-means or the $k$-median problem is to require that facility locations are chosen from the set of input points so that $\Sigma \subseteq S$. However, this problem version for $k$-means is not covered here.

For a given set of facility locations $\Sigma$, let $S_i = \{p \in S \mid ||\sigma_i - p|| \leq ||\sigma_j - p|| \ \forall j\}$, where $S_i \subseteq S$, be the subset of points that are closest to facility location $\sigma_i$ (or at least as close to $\sigma_i$ as to any other facility location). We can assign each of the $n$ points in $S$ to exactly one subset $S_i$, breaking ties in distance arbitrarily. The cost of this choice for our set of facility locations, according to the $k$-means viewpoint, is given as:

$$\mu(S, \Sigma) = \left[ \frac{1}{n} \sum_{i=1}^{k} \sum_{p \in S_i} ||p - \sigma_i||^2 \right]^{1/2}.$$

The optimal choice (lowest cost) for all possible facility location sets is then:

$$\mu(S) = \min_{\Sigma \in \begin{pmatrix} \Re^d \\ k \end{pmatrix}} \mu(S, \Sigma) \; .$$

When actually solving this problem for $k = 1$, if $x$ is a single facility location, the answer becomes:

$$\min_{x} \sum_{i=1}^{n} ||x - p_i||^2 \; .$$

In this case all points were given the same weight or importance, but it would be a simple generalization to place weights on the summation terms so that each of the $n$ points has a different importance.

Returning to the equal-weight problem, consider the case where we are working in $\Re^d$. Then each of our $n$ points may be represented as $p_i = (p_i^{(1)}, \ldots, p_i^{(d)})$, where $p_i^{(j)}$ is the dimension $j$ coordinate of point $p_i$. Similarly, our single facility is $x = (x^{(1)}, \ldots, x^{(d)})$. Now we can restate our formula as follows. Set $f(x)$ as:

$$\begin{aligned} f(x) = \sum_{i=1}^{n} ||x - p_i||^2 &= \sum_{i=1}^{n} \left( ||x||^2 + ||p_i||^2 - 2\langle x, p_i \rangle \right) \\ &= n||x||^2 + \sum_{i=1}^{n} ||p_i||^2 - 2 \left\langle x, \sum_{i=1}^{n} p_i \right\rangle \end{aligned}$$

To minimize $f(x)$, we set $\frac{\partial f}{\partial x^{(j)}} = 0$ for all $j$. That is:

$$\begin{aligned} \frac{\partial f}{\partial x^{(j)}} &= 2nx^{(j)} - 2\sum_{i=1}^{n} p_i^{(j)} = 0 \; , \\ \Rightarrow \quad x^{(j)} &= \frac{1}{n} \sum_{i=1}^{n} p_i^{(j)} \; , \\ \Rightarrow \quad x &= \frac{1}{n} \sum_{i=1}^{n} p_i \; . \end{aligned}$$

Hence, the *centriod* — the center of mass of the $n$ points — minimizes the sum of squares of distances. Thus, for a given set of points we have a method to choose a single optimal facility location for that set. Considering the task of finding $k$ facility locations, if the total set of points $S$ were already divided into $k$ subsets $S_1, \ldots, S_k$ then we could treat each $S_i$ as a unique set where we find a single, optimal facility location. Conversely, given a set of $k$ facility locations $\sigma_1, \ldots, \sigma_k$, it is relatively simple to assign points to the closest facility, forming $k$ subsets $S_i$. This relation is noted below.

| Given: | | Can be used to find: |
|---|---|---|
| $\sigma_1, \ldots, \sigma_k$ | $\rightarrow$ | $S_1, \ldots, S_k$ |
| $S_1, \ldots, S_k$ | $\rightarrow$ | $\sigma_1, \ldots, \sigma_k$ |

From this simple table a possible algorithm for finding $k$ facility locations for a set of $n$ points is apparent: Start with a guess of one side and figure out the other. For example, pick some method to initially place $k$ facilities $\sigma_i$. Using these $\sigma_i$, group the $n$ points into subsets $S_i$ based on their closest facilities. Using these $S_i$, recompute the $\sigma_i$, etc. and continue moving back and forth until convergence. This is the method given in the pseudo-code of Algorithm 1. In that pseudo-code $\sigma_i^j$ denotes facility location $i$ at algorithm iteration $j$ while $\Sigma^j$ gives the total set of $k$ facility locations at iteration $j$. Similarly $S_i^j$ gives the set of input points $p_i \in P$ that are assigned to facility location $\sigma_i$ for algorithm iteration $j$.

**Algorithm 1** *The $k$-means Method*

$\Sigma^0$ : Choose $k$ points in $\Re^d$.
**while** $j = 0$ **or** $\Sigma^j \neq \Sigma^{j-1}$ **do**
  **for** $1 \leq i \leq k$ **do**
    $S_i^{j+1} = \text{Vor}(\sigma_i^j) \cap S$
    $\sigma_i^{j+1} = \text{centroid}(S_i^{j+1})$
  **end for**
  $\Sigma^{j+1} = \cup_{i=1}^{k} \sigma_i^{j+1}$
  $j = j + 1$
**end while**
**return** $\Sigma^j$

In Algorithm 1, $\text{Vor}(\sigma_i^j)$ denotes the **Vornoi Cell** of $\sigma_i^j$ in the **Vornoi Diagram** of $\Sigma^j$. Recall that, given a set of input points $x_i \in X$, the Vornoi Diagram of a space divides that space into non-overlapping cells, one for each input point $x_i$, such that any other point $y \notin X$ that lies in the cell containing $x_i$ is at least as close to $x_i$ as to any other input point $x_j \in X$. (So, a Vornoi Cell around a given $x_i$ is exactly the set of such points $y$ that are closer to $x_i$.) Boundaries between adjacent Vornoi Cells mark points that are equally close to two or more $x_i$'s. In Algorithm 1 above, the current set of facility location points $\Sigma^j$ is the input set $X$ given to create the Vornoi Diagram. For this algorithm we can make the following two claims which are given here without proof.

**Claim 1** ***(Stuard Lloyd, 1982)*** *$\mu(S, \Sigma^{j+1}) \leq \mu(S, \Sigma^j)$ and equality holds if and only if $\Sigma^{j+1} = \Sigma^j$.*

**Claim 2** *The number of distinct partitions $= n^{O(kd)}$ when $\Sigma^{j+1} = \Sigma^j$.*

An interesting problem related to the $k$-means algorithm given is, *What is the proper value for k?* A lot of research has been devoted to developing methods for finding appropriate values for $k$ for different settings, but that is not a topic covered here. For the $k$-means algorithm itself, an $\Omega(n)$ lower bound on the possible running time of the algorithm was the best known result until just last year (2006). Arthur and Vassilvitskii [SoCG '06] (David Arthur is a Duke graduate) proved the following:

**Claim 3** *$\exists$ a set $P$ of $n$ points so that this $k$-means algorithm on $P$ takes $2^{\Omega(\sqrt{n})}$ steps.*

While an exponential lower bound may be troubling, the $k$-means algorithm does well in practice (just like the Simplex Method discussed in earlier course lectures). Once again, a smoothed analysis result helps explain why this occurs. A polynomial number of steps is required to run the $k$-means algorithm in the smoothed analysis viewpoint [Arthur, Vassilvitskii 06].

Recall that smoothed analysis interpolates between the worst case and average case running time for an algorithm. So, this result states that while there may be some problem instances where $k$-means does very

badly, such instances are rare. Also, any slight perturbation of those bad problems will result, with high probability, in a problem instance that can be quickly solved. Papers related to these topics are posted on the course website on the 'Readings' page.

Aside from the running time, there may also be a concern about the possible performance bounds for this $k$-means method. In fact, $k$-means gives a local minimum and there is no guarantee about how good, or poor, the answer might be. One important aspect in determining the performance of this algorithm is how we choose the initial set of $k$ facility locations $\Sigma^0$. One possible method for initially choosing $k$ facility locations $\sigma_1, \ldots, \sigma_k$ will now be given. To begin, choose the first facility location $\sigma_0$ arbitrarily. To choose the remaining $\sigma_i$ locations, we make use of the value:

$$D(x) = \min_i ||x - \sigma_i|| \ .$$

Assume that we have so far picked facility locations $\sigma_0, \ldots, \sigma_i$. We will next choose location $\sigma_{i+1}$. For any given point $p_j$ in the total space where our $\sigma_i$ points are coming from, let point $p_j$ be chosen to be $\sigma_{i+1}$ with the following probability.

$$\text{Prob}\left[\sigma_{i+1} = p_j\right] = \frac{D^2(p_j)}{\sum_{j=1}^{n} D^2(p_j)}$$

Given that we choose our $k$ facility locations in this manner, we can bound the expected performance of this method. Let $\mu_{\text{opt}}$ be the optimal cost value possible for assigning $k$ facility locations for some set of $n$ input points. Then we may bound the expected cost $\mu$ given by our method.

**Claim 4** $E[\mu] = \Omega(\log k)\mu_{opt}$ *or, more precisely,* $E[\mu] \leq 8(\ln k + 1)\mu_{opt}$.

### 15.1.2 The $k$-median Problem

The $k$-median problem gives another interpretation for how the $n$ input points can be "close" to the $k$ facility locations we choose. In this problem the goal is to minimize the average distance between input points and the facility locations to which they are assigned. Once again, let $S$ be the set of input points in $\Re^d$, $\Sigma$ be the set of $k$ chosen facility locations, and $S_i \subseteq S$ the portion of the input points assigned to facility location $\sigma_i$. The cost of a given set of facility locations is given as:

$$\text{Cost:} \quad \nu(S, \Sigma) = \frac{1}{n} \sum_{i=1}^{k} \sum_{p \in S_i} ||p - \sigma_i|| \ .$$

The goal is of course to minimize this cost, which may be stated as:

$$\text{Goal:} \quad \min_{\Sigma \in \begin{pmatrix} \Re^d \\ k \end{pmatrix}} \nu(S, \Sigma) \ .$$

Finding an optimal solution to the $k$-median problem is NP-hard just as the $k$-means problem was. Thus, other methods have been developed for giving approximate solutions. A paper in SODA '07 by Ke Chen addresses the topic of finding coresets for $k$-median. Recall that a **coreset** is a subset of the input points such that the subset approximates the original input, according to some measurement, within a certain error level. Of course it is desired that this coreset be small while still allowing for a good approximation.

Consider a weighted version of the $k$-median problem where we a weight function $w : S \to \Re^+$. That is, $w(p)$ assigns positive, real values to input points $p \in S$. Now the cost for a set of facility locations $\Sigma$ is given by:

$$\mu(S, \Sigma) = \frac{1}{n} \sum_{i=1}^{n} \sum_{p \in S_i} ||p - \sigma_i|| \, w(p) \; .$$

Let $C \subseteq S$ be a chosen subset of the input points and $\epsilon \in \Re^+$ some positive number. Along with the weight function, we say that $C, w$ is a $(k, \epsilon)$-**coreset for** $k$-**median** if:

$$\forall \, \Sigma \;\; |\mu(C, \Sigma) - \mu(S, \Sigma)| \leq \epsilon \cdot \mu(S, \Sigma) \; .$$

Ke Chen [SODA '07] proves the following:

**Claim 5** *Given a set $P$ of $n$ points in $\Re^d$, there exists a $(k, \epsilon)$-coreset of $P$ of size $O(\frac{dk^2}{\epsilon^2} \log \frac{n}{\epsilon})$.*

For another approximation, result Arora, et. al. showed a method for finding a $(1 + \epsilon)$-approximation for $k$-median in $n^{O(1/\epsilon)}$ time. Furthering this work, Rao et. al. gave an approximation result that ran in time $\exp\left(\frac{1}{\epsilon^{O(1)}}\right) n \log^{O(1)} n$, where the facility locations are chosen among the inputs points in $S$.

## 15.2 LP Relaxation for $k$-Median

As with many other optimization problems, $k$-median is a special case of integer programming. For this problem let our set of facility locations be chosen from the given set of input points. To express $k$-median in this representation let $p_1, \ldots, p_n \in P$ again be our $n$ input points. Now let variables $c_{ij} = ||p_i - p_j||$, $1 \leq i, j \leq n$, and assume we have $y_1, \ldots, y_n$ where each $y_i \in \{0, 1\}$. Variable $y_j = 1$ if point $p_j \in P$ is chosen as one of the $k$ facilities, and $y_j = 0$ otherwise. Also set variables $x_{ij} \in \{0, 1\}$ with $x_{ij} = 1$ if point $i$ is assigned to cluster $j$ and $x_{ij} = 0$ otherwise. The integer programming representation of the $k$-median problem can now be given.

Integer Program

| | | | |
|---|---|---|---|
| minimize: | $\sum_{i,j} c_{ij} x_{ij}$ | | |
| such that: | | | |
| | $\sum_{j=1}^{n} x_{ij}$ | = | 1 |
| | $\sum_{j=1}^{n} y_j$ | $\leq$ | $k$ |
| | $x_{ij}$ | $\leq$ | $y_j$ |

The program above solves for the case where each input point $p_i$ is treated equally. As another alternative, we could weight our points with a function $w$. The integer program would then be minimizing the function $\sum_{i,j} c_{ij}x_{ij} + \sum_{i,j} wx_{ij}$ .

While the integer programming representation gives an exact solution to the $k$-median problem, this is still an NP-hard problem. However, a linear programming relaxation is possible and this does achieve a tractable, but approximate, solution. For this case we simply take the integer program given earlier and relax the variable requirements so that $x_{ij} \in [0, 1]$ and $y_i \in [0, 1]$ rather than making a hard, 0 or 1 decision. This LP gives a fractional solution: $\hat{x}_{ij} \cdot \hat{y}_j$. Since it is often necessary to solve the problem without fractions, this leaves the question: *How do we round to an integer solution?* Several ideas have been proposed for this task. One such idea is to round up the $k$ top $y_j$'s with a value closest to 1 (since each $\hat{y_j} \leq 1$). Another idea — one which will be used here — is to randomly chose $k$ of the $y_j$'s where each is chosen with probability $\hat{y_j}$. That is,

$$\Pr[\ p_j \text{ is chosen as a center (facility) }] = \hat{y_j}.$$

With this method of rounding we can make some guarantees about the quality of our chosen solution. Let $c_{opt}$ be the cost of the optimal solution (set of $k$ facilities) for a given $k$-median problem. Given the linear programming fractional solution to this problem we can choose slightly more than $k$ facilities and then bound the performance of the resulting solution set. Specifically, if we choose $(1 + \frac{1}{\epsilon})k \ln \frac{n}{\delta}$ points to round up and become full facilities then the resulting cost of this solution is $(1+\epsilon) \cdot c_{\text{opt}}$ with probability $\geq 1 - \delta$. In these formulas $\epsilon, \delta \in \Re^+$.

To see how this is possible, consider some point $p_i$ from the set initially given as input to the problem. Set $\hat{c}_i = \sum_{j=1}^{n} c_{ij}\hat{x}_{ij}$. Recall that $\hat{x}_{ij}$ is the fraction of point $p_i$ that is assigned to facility point $p_j$. Thus, $\hat{c}_i$ is the total cost of fractionally assigning point $p_i$ to each of those facilities. If we instead assign point $p_i$ to a single facility, which is our goal, then we can think of $\hat{c}_i$ as the radius of a circle surrounding $p_i$.

In Figure 15.2 the red circle with radius $\hat{c}_i$ shows the area where we could find a single facility point $p_j$ where assigning $p_i$ to that facility will cost at most $\hat{c}_i$. Similarly, the larger circle with radius $(1 + \epsilon)\hat{c}_i$ gives an area where we can find a single facility with a cost that gets sufficiently close to $\hat{c}_i$. In this problem the facility locations are chosen from among the set of input points $P$, so we really care about those points $p_j \in P$ that lie within the larger circle shown. This set of points is given as:

$$V_i : \{p_j \,|\, ||p_i - p_j|| \leq (1 + \epsilon)\hat{c}_i\} .$$

Thus, $V_i$ is precisely the set of points $p_j$ that would make acceptable facility locations for point $p_i$ to be assigned to. Now, for the task of transforming our linear programming solution into a proper choice of facilities, choose $s$ points from set $P$ at random to form our set of facilities: $Q = \{q_1, \ldots, q_s\}$. The probability with which each point $p_i \in P$ is chosen and the total size of the chosen set, $s = |Q|$, are both given below.

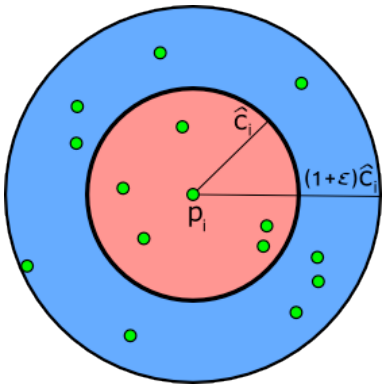


Figure 15.2: Two circles denote regions where acceptably close facility locations may lie.

$$\Pr[p_i \text{ is chosen for } Q] = \frac{\hat{y}_i}{\sum_{i=1}^{n} \hat{y}_i}$$

$$s = \left(1 + \frac{1}{\epsilon}\right) \cdot k \cdot \ln\left(\frac{n}{\delta}\right)$$

Given these two values, we may now make two claims about the quality of our chosen set of facilities.

**Claim 6** *With probability at least* $1 - \delta$, $Q \cap V_i \neq \emptyset$ *for every* $i$.

**Claim 7** $\Pr[$*A random point lies in* $V_i] \geq \frac{\epsilon}{(\epsilon+1)k}$.

First, Claim 6 states that with a certain probability all points $p_i$ will have at least one facility location $q_j \in Q$ that is acceptably close. Any such $q_j$ is acceptably close if it is contained within $V_i$. For the second part, Claim 7 gives the chance of any chosen point falling inside set $V_i$ — such as a point chosen to be a facility location. An interesting relationship between these two statements is that Claim 7 implies Claim 6. To see this, first consider that the probability of a random point not falling within set $V_i$ is 1 minus the probability given in Claim 7. From this we can derive Claim 6.

$$\begin{aligned}
\Pr[\text{ A random point} \notin V_i] &= \left(1 - \frac{\epsilon}{(\epsilon+1)k}\right) \\
\Rightarrow \quad \Pr[\ V_i \cap Q = \emptyset\ ] &= \left(1 - \frac{\epsilon}{(1+\epsilon)k}\right)^s \\
&\leq \exp\left(\frac{-\epsilon}{(1+\epsilon)k} \cdot s\right) \\
&= \frac{\delta}{n}\,.
\end{aligned}$$

Considering if any of the $n$ such $V_i$ sets did not contain a facility location point means multiplying together the probability of each such event. Hence,

$$\Pr[\ \exists i \text{ such that } V_i \cap Q = \emptyset\ ] \leq \delta\,.$$

Since this is the opposite event of the one given in Claim 6, then 1 minus the probability above gives the chance that every point $p_i$ has a facility location acceptably close. Thus $1 - \delta$ is the probability of our algorithm giving a suitable approximate solution, just as stated in Claim 6.