CSEP 521: Dimension Reduction

Richard Anderson, March 2, 2021

Announcements

- Homework 9 is available, Due March 11.
- Course grade based on top 8 of 9 homeworks. All weighted equally.
- Remaining lectures
  - Optimization to linear programming and beyond

Dimension Reduction

- A key problem in working with large scale data sets
  - Can we reduce representation size at the expense of a small error

Warmup for dimension reduction for R^n

- Consider the distance function D(x,y) = 0 if x = y, D(x,y) = 1 if $x \neq y$
- Suppose we have a domain U and want to answer distance queries between a set of n elements
- Natural solution is to use $\log_2 U$ bits to describe the elements
- Can we use less space if we want to approximately answer distance queries

Scenario: We have a database of objects (e.g. people), and a field with domain U (e.g. favorite movie) – we are interested in reducing the space required to store the field info and to answer the query of whether or not X and Y have the same favorite movie.

Of course this is going to be hashing

- Choose a good hash function h: $U \rightarrow 2^{32}$
- Let $f_1(x) = h(x) \bmod 2$
- 1 bit representation
  - If $x = y$, then $f_1(x) = f_1(y)$
  - If $x \neq y$, then $\mathbf{Pr}[f_1(x) = f_1(y)] \leq ½$
  - Property preserved with probability at least 50%
- Repeat with k independent has functions $h_1, \ldots, h_k$
  - If $x = y$, then $f_i(x) = f_i(y)$ for all $i = 1, \ldots, k$
  - If $x \neq y$, then $\mathbf{Pr}[f_i(x) = f_i(y) \text{ for all } i = 1, \ldots, k] \leq 2^{-k}$
- To achieve error of δ, we need to use $k = \lceil \log_2 1/\delta \rceil$

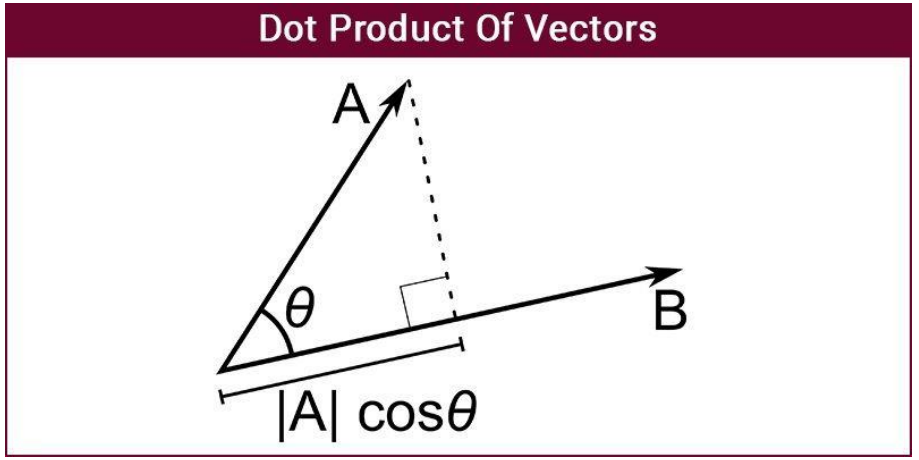
Random Projections for L_2 Distance in R^N

- Johnson-Lindenstrauss Transform
  - Extensions of Lipschitz mappings into a Hilbert space, William Johnson and Joram Lindenstrauss, 1984
  - Pure mathematics result that crossed over to Computer Science

- Project from R^N to a random R^K dimensional subspace where K is $O(\varepsilon^{-2} \log N)$ and distances are preserved to a factor of $1+\varepsilon$
  - In practice, $K \approx 100$

Projections

- Projection onto a line
- Inner Product
- If b is a unit vector than a · b gives the position of a when projected onto b
- Project [4, -1, 3] onto [2, 1, 1]

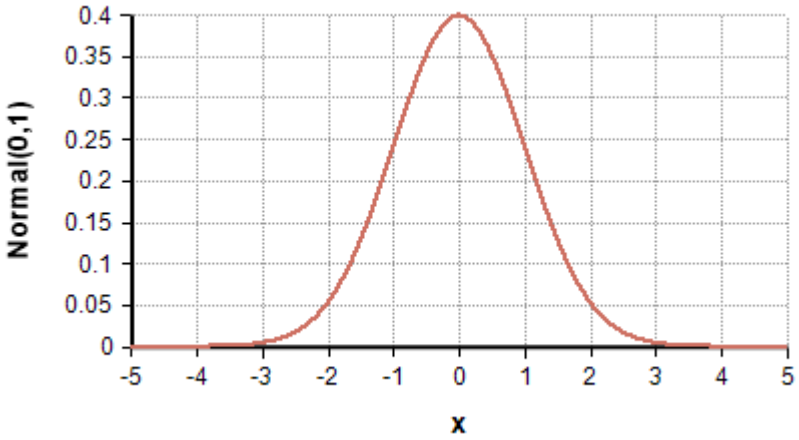


$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos\theta,$$

Gaussian Distribution

- $N(\mu,\sigma^2)$ – Normal distribution with mean μ and variance σ^2
- X_1 and X_2 are independent random variables with distribution $N(\mu_1,{\sigma_1}^2)$ and $N(\mu_2,{\sigma_2}^2)$ then $X_1 + X_2$ has distribution $N(\mu_1+ \mu_2,{\sigma_1}^2+ {\sigma_2}^2)$



$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

$\mu = 0$ and $\sigma = 1$

$$\varphi(x) = \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}}$$

Aside: Generating Gaussians

- Box-Muller methods generates a pair of independent Gaussian RVs from two random points from [0,1)
- Mapping of unit square to independent Gaussians
- Many languages have a Gaussian generator (Matlab, Python, Java)
- Web sites will generate them for you

Suppose U_1 and U_2 are independent samples chosen from the uniform distribution on the unit interval (0, 1). Let

$$Z_0 = \sqrt{-2\ln U_1}\cos(2\pi U_2)$$

and

$$Z_1 = \sqrt{-2\ln U_1}\sin(2\pi U_2).$$

Then Z_0 and Z_1 are independent random variables with a standard normal distribution.

Random Projection

- Objects are points in n dimensional Euclidean space R^n
- Choose random vector $r = (r_1, \ldots, r_n) \in R^n$
- Real valued function $f_r : R^n \rightarrow R$

$$f_{\mathbf{r}}(\mathbf{x}) = \langle \mathbf{x}, \mathbf{r} \rangle = \sum_{j=1}^{n} x_j r_j$$

- Random linear combination of components of x

Unbiased Estimator of Squared L_2 Distance

- For x, y in R^n, $(f_r(x) - f_r(y))^2$ is an unbiased estimator of $|x - y|^2$
- Fix x, y in R^n

$$f_{\mathbf{r}}(\mathbf{x}) - f_{\mathbf{r}}(\mathbf{y}) = \sum_{j=1}^{n} x_j r_j - \sum_{j=1}^{n} y_j r_j = \sum_{j=1}^{n} (x_j - y_j) r_j$$

- r_j is a Gaussian with mean zero and variance 1, $(x_j - y_j)r_j$ is a Gaussian with mean zero and variance $(x_j - y_j)^2$
- Right-hand side is a Gaussian with mean zero and variance

$$\sum_{j=1}^{n} (x_j - y_j)^2 = \|\mathbf{x} - \mathbf{y}\|_2^2$$

Cont.

- By definition $Var(X) = E((X - E(X])^2)$, so $Var(X) = E(X^2)$ when $E(X) = 0$
- Taking X as the random variable $f_r(x) - f_r(y)$

$$\mathbf{E}\left[(f_{\mathbf{r}}(\mathbf{x}) - f_{\mathbf{r}}(\mathbf{y}))^2\right] = \|\mathbf{x} - \mathbf{y}\|_2^2$$

- Estimator of the squared L_2 distance between x and y

Independent Trials

- Pick d vectors $r_1, \ldots, r_d$
- Each vector is drawn i.i.d. from a standard Gaussian
- Given points x, y, we get d independent estimates of $|x-y|^2$
- "One can figure out exactly how large d needs to be to achieve a target approximation''

For a set of k points in n dimensions, to preserve all k(k-1)/2 interpoint Euclidean distances up to a $1 \pm \varepsilon$ factor, set $d = \Theta(\varepsilon^{-2} \log k)$

Johnson-Lindenstrauss Transform

- JL transform maps from R^n to R^d, where d is selected based on desired accuracy
- The JL transform is represented as a $d \times n$ matrix A where each of the dn entries is chosen i.i.d. from a standard Gaussian distribution
- Mapping from n vectors to d vectors is defined $x \rightarrow Ax$ / sqrt(d)
- 1/sqrt(d) factor scales to be an average over d estimates

Simplification

- D. Achlioptas (2003). Similar results hold if matrix entries are chosen uniformly from {-1, 1} or from {-sqrt(3), 0, sqrt(3)} with probability 1/6, 2/3, 1/6 respectively
- Proof:

Proof of Lemma 5.1. We start with the upper tail. For arbitrary $h>0$ let us write

$$\Pr\left[S>(1+\varepsilon)\frac{k}{d}\right]=\Pr\left[\exp(hS)>\exp\left(h(1+\varepsilon)\frac{k}{d}\right)\right]$$
$$<\mathbf{E}(\exp(hS))\exp\left(-h(1+\varepsilon)\frac{k}{d}\right).$$

Since $\{Q_j\}_{j=1}^k$ are i.i.d. we have

$$\mathbf{E}(\exp(hS))=\mathbf{E}\left(\prod_{j=1}^{k}\exp(hQ_j^2)\right) \tag{8}$$
$$=\prod_{j=1}^{k}\mathbf{E}(\exp(hQ_j^2)) \tag{9}$$
$$=(\mathbf{E}(\exp(hQ_1^2)))^k, \tag{10}$$

where passing from (8) to (9) uses that the $\{Q_j\}_{j=1}^k$ are independent, while passing from (9) to (10) uses that they are identically distributed. Thus, for any $\varepsilon>0$

$$\Pr\left[S>(1+\varepsilon)\frac{k}{d}\right]<(\mathbf{E}(\exp(hQ_1^2)))^k\exp\left(-h(1+\varepsilon)\frac{k}{d}\right). \tag{11}$$

Substituting (6) in (11) we get (12). To optimize the bound we set the derivative in (12) with respect to h to 0. This gives $h=\frac{d}{2}\frac{\varepsilon}{1+\varepsilon}<\frac{d}{2}$. Substituting this value of h we get (13) and series expansion yields (14).

$$\Pr\left[S>(1+\varepsilon)\frac{k}{d}\right]<\left(\frac{1}{\sqrt{1-2h/d}}\right)^k\exp\left(-h(1+\varepsilon)\frac{k}{d}\right) \tag{12}$$
$$=((1+\varepsilon)\exp(-\varepsilon))^{k/2} \tag{13}$$
$$<\exp\left(-\frac{k}{2}(\varepsilon^2/2-\varepsilon^3/3)\right). \tag{14}$$

Proof of Lemma 5.2. To prove (7) we observe that for any unit vector α, by (20) and (21),

$$\mathbf{E}(Q(\alpha)^4)\leqslant\mathbf{E}(Q(w)^4)\leqslant\mathbf{E}(T^4),$$

while

$$\mathbf{E}(T^4)=\int_{-\infty}^{+\infty}\frac{1}{\sqrt{2\pi}}\exp(-\lambda^2/2)\left(\frac{\lambda^4}{d^2}\right)d\lambda=\frac{3}{d^2}.$$

To prove (6) we first observe that for any real-valued random variable U and for all h such that $\mathbf{E}(\exp(hU^2))$ is bounded, the Monotone Convergence Theorem (MCT) allows us to swap the expectation with the sum and get

$$\mathbf{E}(\exp(hU^2))=\mathbf{E}\left(\sum_{k=0}^{\infty}\frac{(hU^2)^k}{k!}\right)=\sum_{k=0}^{\infty}\frac{h^k}{k!}\mathbf{E}(U^{2k}).$$

So, below, we proceed as follows. Taking $h\in[0,d/2)$ makes the integral in (22) converge, giving us (23). Thus, for such h, we can apply the MCT to get (24). Now, applying (20) and (21)–(24) gives (25). Applying the MCT once more gives (26).

$$\mathbf{E}(\exp(hT^2))=\int_{-\infty}^{+\infty}\frac{1}{\sqrt{2\pi}}\exp(-\lambda^2/2)\exp\left(h\frac{\lambda^2}{d}\right)d\lambda \tag{22}$$
$$=\frac{1}{\sqrt{1-2h/d}} \tag{23}$$
$$=\sum_{k=0}^{\infty}\frac{h^k}{k!}\mathbf{E}(T^{2k}) \tag{24}$$
$$\geqslant\sum_{k=0}^{\infty}\frac{h^k}{k!}\mathbf{E}(Q(\alpha)^{2k}) \tag{25}$$
$$=\mathbf{E}(\exp(hQ(\alpha)^2)). \tag{26}$$

Thus, $\mathbf{E}(\exp(hQ^2))\leqslant 1/\sqrt{1-2h/d}$ for $h\in[0,d/2)$, as desired. □

Lemma 6.3. *Let r_1, r_2 be i.i.d. random variables having one of the two probability distributions given by Eqs. (1) and (2) in Theorem 1.1.*
For any $a,b\in\mathbb{R}$ let $c=\sqrt{(a^2+b^2)/2}$. Then for any $M\in\mathbb{R}$ and all $k=0,1,\ldots$

$$\mathbf{E}((M+ar_1+br_2)^{2k})\leqslant\mathbf{E}((M+cr_1+cr_2)^{2k}).$$

Proof. We first consider the case where $r_i\in\{-1,+1\}$.
If $a^2=b^2$ then $a=c$ and the lemma holds with equality. Otherwise, observe that

$$\mathbf{E}((M+cr_1+cr_2)^{2k})-\mathbf{E}((M+ar_1+br_2)^{2k})=\frac{S_k}{4},$$

where

$$S_k=(M+2c)^{2k}+2M^{2k}+(M-2c)^{2k}-(M+a+b)^{2k}$$
$$-(M+a-b)^{2k}-(M-a+b)^{2k}-(M-a-b)^{2k}.$$

We will show that $S_k\geqslant 0$ for all $k\geqslant 0$.
Since $a^2\neq b^2$ we can use the binomial theorem to expand every term other than $2M^{2k}$ in S_k and get

$$S_k=2M^{2k}+\sum_{i=0}^{2k}\binom{2k}{i}M^{2k-i}D_i,$$

where

$$D_i=(2c)^i+(-2c)^i-(a+b)^i-(a-b)^i-(-a+b)^i-(-a-b)^i.$$

Observe now that for odd i, $D_i=0$. Moreover, we claim that $D_{2j}\geqslant 0$ for all $j\geqslant 1$. To see this claim observe that $(2a^2+2b^2)=(a+b)^2+(a-b)^2$ and that for all $j\geqslant 1$ and $x,y\geqslant 0$, $(x+y)^j\geqslant x^j+y^j$. Thus, $(2c)^{2j}=(2a^2+2b^2)^j=[(a+b)^2+(a-b)^2]^j\geqslant(a+b)^{2j}+(a-b)^{2j}$ implying

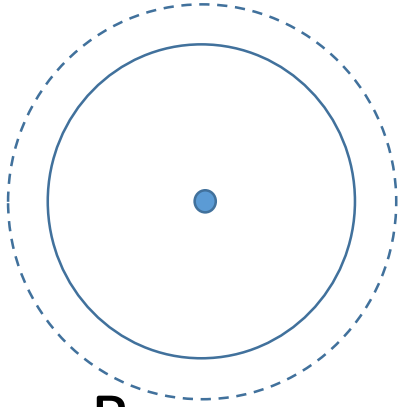
$$S_k=2M^{2k}+\sum_{j=0}^{k}\binom{2k}{2j}M^{2(k-j)}D_{2j}=\sum_{j=1}^{k}\binom{2k}{2j}M^{2(k-j)}D_{2j}\geqslant 0.$$

The proof for the case where $r_i\in\{-\sqrt{3},0,+\sqrt{3}\}$ is just a more cumbersome version of the proof above, so we omit it. That proof, though, brings forward an interesting point. If one tries to take $r_i=0$ with probability greater than 2/3, while maintaining that r_i has a range of size 3 and variance 1, the lemma fails. In other words, 2/3 is tight in terms of how much probability mass we can put to $r_i=0$ and still have the current lemma hold. □

Applications

- High dimensional data sets
  - Facebook
    - Friend neighborhood
    - Stories followed
  - Biochemistry
    - Candidate compounds for pharmaceuticals
  - Youtube
    - Videos watched
  - Phone metadata
    - Numbers called

Range queries

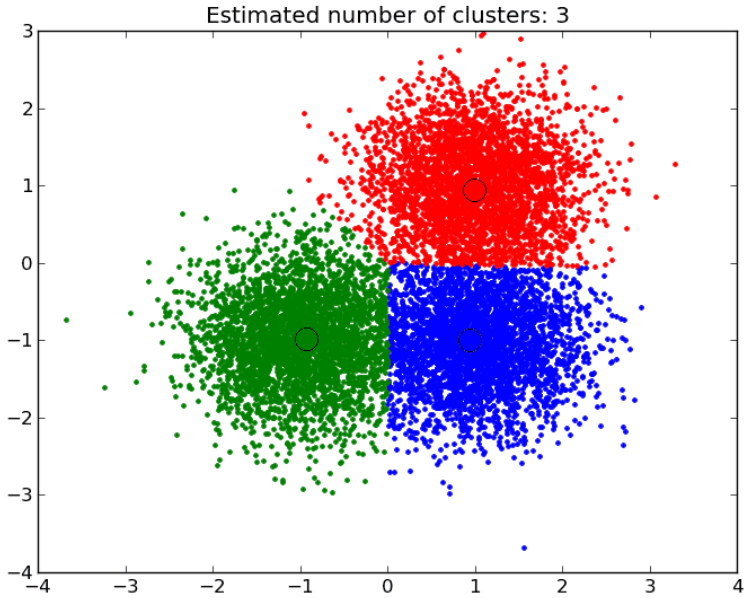


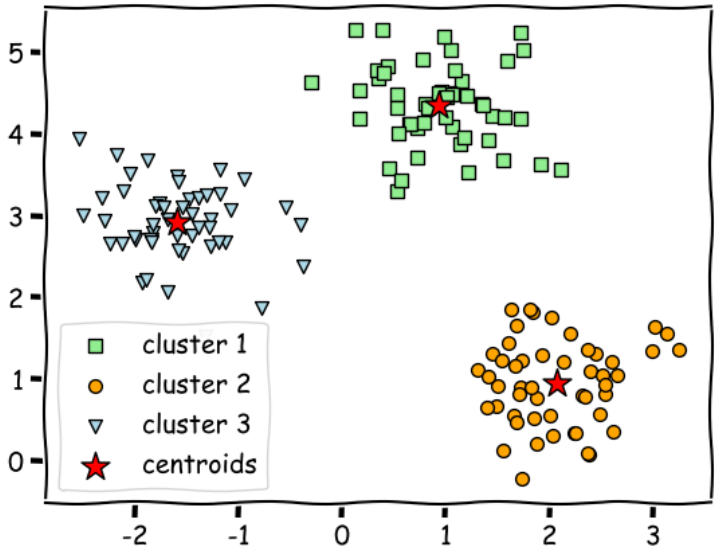
- Queries such as “k-Nearest Neighbors”, all points within distance B, count points within distance B
- Standard approach – reduce dimension and search using k-D trees*
- Results subject to errors by factors of $1 \pm \varepsilon$
- k-Nearest Neighbors
  - Given query point y, return k points within $(1+\varepsilon)B$ of y, where B is the k-NN distance from y
- Points with distance B
  - Given query point y, return a set of points which contains all points within distance $(1- \varepsilon)B$ of y and no point of distance greater than $(1+ \varepsilon)B$ of y

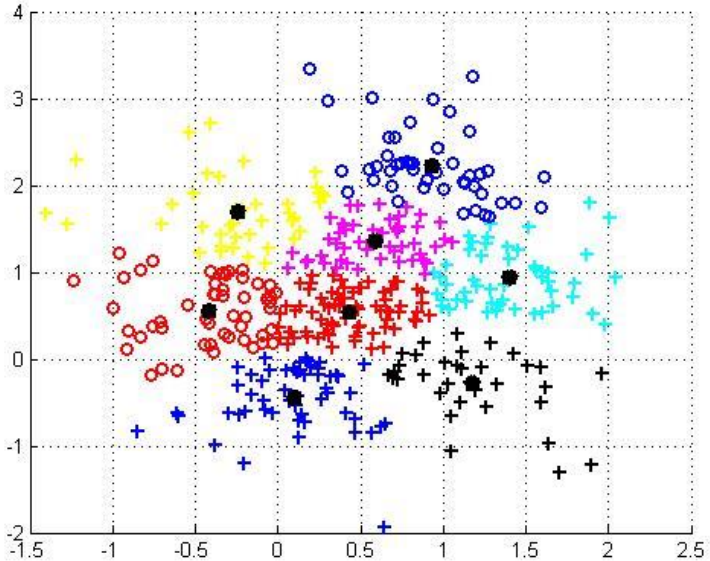
* Different k

k-means clustering

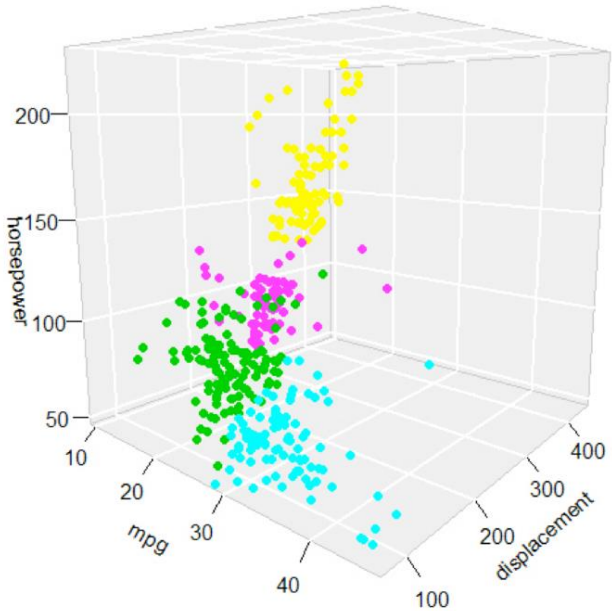
- Given S, a set of n points in R^m, find k representative points in R^m that that best partition the data into clusters
- Application – use these points for classification – a new point finds it nearest neighbor among the k points. Rocchio Algorithm.
- Partition the space into Voronoi cells
- k-clustering of S is a partition into k subsets
  - The cluster-variance is the sum of the squares of the distances of each point to the respective center of its cluster
- The k-means clustering of a set S is the k-clustering that minimizes the cluster-variance
- Finding the optimal k-means cluster is NP-Hard, but we will ignore that







Clustering of Horsepower, MPG, and Displacement



Higher dimensional clustering

- It gets harder to draw!
- Same idea works in high dimensions

- Results show that k-means clustering can be done after dimension reduction, greatly improving performance of constructing and using clustering (at the expense of 1+ε error)

- Heuristic algorithms are used to construct a k-means clustering (with common confusion between the algorithm and the definition of clustering)

Lloyd's Algorithm (Stuart Lloyd, Bell Labs, 1957)

- Iterative algorithm that (usually) converges to a good approximation.

- Pick an initial clustering
- Repeat until tired
  - Compute centers of clusters
  - Reassign points to closest center

```
for (int i = 0; i < N; i++){
   CS.AddPoint(i, i % K);
}

int count = 0;
while (count < TOO_LONG){
   CS.SetCenters();

   bool moved = false;
   for (int i = 0; i < N; i++){
      int g = CS.Group(i);
      double d_min = dist(P[i], CS.Center(g));
      for (j = 0; j < K; j++){
         if (j == CS.Group(i))
            continue;
         double d = dist(P[i], CS.Center(g));
         if (d < d_min){
            g = j;  d_min = d;
         }
      }
      if (g != CS.Group(i)){
         CS.MovePoint(i, g);
         moved = true;
      }
   }
   count++;
   if (! moved)
      break;
}
```

Lloyd's Algorithm

CS is a ClusterSet which associates points with clusters and maintains the centers of the clusters.

Methods

AddPoint(int i, int g);
MovePoint(int i, int g);
SetCenters();
Group(int i);
Center(int g);