

## 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 $\mathbb{R}^n$

- 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

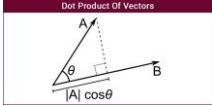


- Choose a good hash function  $h: U \rightarrow 2^{32}$
- Let  $f_1(x) = h(x) \bmod 2$
- 1 bit representation
  - If  $x = y$ , then  $f_1(x) = f_1(y)$
  - If  $x \neq y$ , then  $\Pr[f_1(x) = f_1(y)] \leq 1/2$
  - Property preserved with probability at least 50%
- Repeat with  $k$  independent hash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

### Random Projections for $L_2$ Distance in $\mathbb{R}^N$

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

### Projections

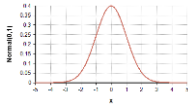


- Projection onto a line
- Inner Product
- If  $b$  is a unit vector then  $a \cdot b$  gives the position of  $a$  when projected onto  $b$
- Project  $[4, -1, 3]$  onto  $[2, 1, 1]$

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

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

### Gaussian Distribution




- $N(\mu, \sigma^2)$  – Normal distribution with mean  $\mu$  and variance  $\sigma^2$
- $X_1$  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  $\sigma = 1 \quad \varphi(x) = \frac{e^{-x^2}}{\sqrt{2\pi}}$

### Aside: Generating Gaussians



- Box-Muller method 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  $\mathbb{R}^n$
- Choose random vector  $r = (r_1, \dots, r_n) \in \mathbb{R}^n$
- Real valued function  $f_r : \mathbb{R}^n \rightarrow \mathbb{R}$

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

- Random linear combination of components of  $\mathbf{x}$

### Unbiased Estimator of Squared $L_2$ Distance

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

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

- $r_j$  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  $\text{Var}(X) = E((X - E(X))^2)$ , so  $\text{Var}(X) = E(X^2)$  when  $E(X) = 0$
- Taking  $X$  as the random variable  $f_r(\mathbf{x}) - f_r(\mathbf{y})$

$$E[(f_r(\mathbf{x}) - f_r(\mathbf{y}))^2] = \|\mathbf{x} - \mathbf{y}\|_2^2$$

- Estimator of the squared  $L_2$  distance between  $\mathbf{x}$  and  $\mathbf{y}$

### Independent Trials

- Pick  $d$  vectors  $r_1, \dots, r_d$
- 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  $1 \pm \epsilon$  factor, set  $d = \Theta(\epsilon^{-2} \log k)$

### 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 1.1. We will use the following lemma, which is proved in the appendix.

Lemma 1.1. Let  $x, y \in R^n$  be two vectors. Let  $r_1, \dots, r_d$  be  $d$  independent standard Gaussian vectors in  $R^n$ . Then

$$\left| \frac{1}{d} \sum_{i=1}^d (x \cdot r_i)^2 - \frac{1}{2} \|x\|^2 \right| \leq \sqrt{\frac{2}{d}} \|x\|$$

Proof. Let  $Z_i = (x \cdot r_i)^2 - \frac{1}{2} \|x\|^2$ . Then  $Z_i$  are independent random variables with  $E[Z_i] = 0$  and  $E[Z_i^2] = \frac{1}{2} \|x\|^4$ . By Chebyshev's inequality,  $P(|\sum Z_i| \geq t) \leq \frac{d E[Z_i^2]}{t^2} = \frac{d \|x\|^4}{2 t^2}$ . Setting  $t = \sqrt{2d} \|x\|^2$  gives the result.

Proof of Lemma 1.2. We will use the following lemma, which is proved in the appendix.

Lemma 1.2. Let  $x, y \in R^n$  be two vectors. Let  $r_1, \dots, r_d$  be  $d$  independent random vectors in  $R^n$  whose entries are chosen uniformly from  $\{-1, 1\}$ . Then

$$\left| \frac{1}{d} \sum_{i=1}^d (x \cdot r_i)^2 - \frac{1}{2} \|x\|^2 \right| \leq \sqrt{\frac{2}{d}} \|x\|$$

Proof. Let  $Z_i = (x \cdot r_i)^2 - \frac{1}{2} \|x\|^2$ . Then  $Z_i$  are independent random variables with  $E[Z_i] = 0$  and  $E[Z_i^2] = \frac{1}{2} \|x\|^4$ . By Chebyshev's inequality,  $P(|\sum Z_i| \geq t) \leq \frac{d E[Z_i^2]}{t^2} = \frac{d \|x\|^4}{2 t^2}$ . Setting  $t = \sqrt{2d} \|x\|^2$  gives the result.

Proof of Lemma 1.3. We will use the following lemma, which is proved in the appendix.

Lemma 1.3. Let  $x, y \in R^n$  be two vectors. Let  $r_1, \dots, r_d$  be  $d$  independent random vectors in  $R^n$  whose entries are chosen uniformly from  $\{-\sqrt{3}, 0, \sqrt{3}\}$  with probability  $1/6, 2/3, 1/6$  respectively. Then

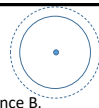
$$\left| \frac{1}{d} \sum_{i=1}^d (x \cdot r_i)^2 - \frac{1}{2} \|x\|^2 \right| \leq \sqrt{\frac{2}{d}} \|x\|$$

Proof. Let  $Z_i = (x \cdot r_i)^2 - \frac{1}{2} \|x\|^2$ . Then  $Z_i$  are independent random variables with  $E[Z_i] = 0$  and  $E[Z_i^2] = \frac{1}{2} \|x\|^4$ . By Chebyshev's inequality,  $P(|\sum Z_i| \geq t) \leq \frac{d E[Z_i^2]}{t^2} = \frac{d \|x\|^4}{2 t^2}$ . Setting  $t = \sqrt{2d} \|x\|^2$  gives the result.

### Applications

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

### Range queries

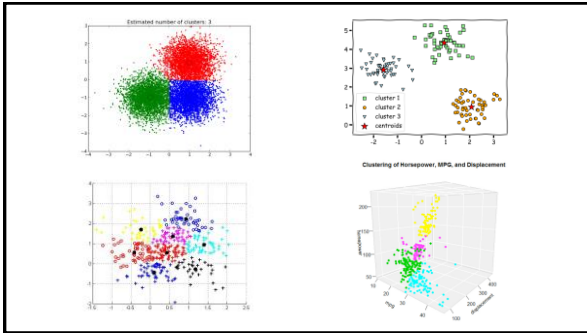


- Queries such as "k-Nearest Neighbors", all points within distance  $B$ , count points within distance  $B$
- Standard approach – reduce dimension and search using  $k$ -D trees\*
- Results subject to errors by factors of  $1 \pm \epsilon$
- k-Nearest Neighbors
  - Given query point  $y$ , return  $k$  points within  $(1+\epsilon)B$  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  $(1-\epsilon)B$  of  $y$  and no point of distance greater than  $(1+\epsilon)B$  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 its 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



## 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  $1+\epsilon$  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(i);

  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);

```