Dimension Reduction- two methods

- Dealing with high d not efficient. Will be nice if we can reduce dimensions.

Dimension Reduction- two methods

- Dealing with high d not efficient. Will be nice if we can reduce dimensions.
- Two Main methods of dimension reduction:

Dimension Reduction- two methods

- Dealing with high d not efficient. Will be nice if we can reduce dimensions.
- Two Main methods of dimension reduction:
- Random Projections: Oblivious to data. [This lecture.]

Dimension Reduction- two methods

- Dealing with high d not efficient. Will be nice if we can reduce dimensions.
- Two Main methods of dimension reduction:
- Random Projections: Oblivious to data. [This lecture.]
  - Can be proved to (a) preserve EVERY length to $1 \pm \varepsilon$, (b) essentially $\varepsilon \in \Omega(1)$.

Dimension Reduction- two methods

- Dealing with high *d* not efficient. Will be nice if we can reduce dimensions.
- Two Main methods of dimension reduction:
- Random Projections: Oblivious to data. [This lecture.]
  - Can be proved to (a) preserve EVERY length to $1 \pm \varepsilon$, (b) essentially $\varepsilon \in \Omega(1)$.
  - Used extensively in theory, because of (a). But (b) is a bottleneck for use in practice.

Dimension Reduction- two methods

- Dealing with high d not efficient. Will be nice if we can reduce dimensions.
- Two Main methods of dimension reduction:
- Random Projections: Oblivious to data. [This lecture.]
  - Can be proved to (a) preserve EVERY length to $1 \pm \varepsilon$, (b) essentially $\varepsilon \in \Omega(1)$.
  - Used extensively in theory, because of (a). But (b) is a bottleneck for use in practice.
- Principal Component Analysis (PCA)

Dimension Reduction- two methods

- Dealing with high d not efficient. Will be nice if we can reduce dimensions.
- Two Main methods of dimension reduction:
- Random Projections: Oblivious to data. [This lecture.]
    - Can be proved to (a) preserve EVERY length to $1 \pm \varepsilon$, (b) essentially $\varepsilon \in \Omega(1)$.
    - Used extensively in theory, because of (a). But (b) is a bottleneck for use in practice.
- Principal Component Analysis (PCA)
    - Projection length behaves very well (better than $\Omega(1)$ error) when averaged over all data, but not guaranteed on each piece.

Dimension Reduction- two methods

- Dealing with high d not efficient. Will be nice if we can reduce dimensions.
- Two Main methods of dimension reduction:
- Random Projections: Oblivious to data. [This lecture.]
  - Can be proved to (a) preserve EVERY length to $1 \pm \varepsilon$, (b) essentially $\varepsilon \in \Omega(1)$.
  - Used extensively in theory, because of (a). But (b) is a bottleneck for use in practice.
- Principal Component Analysis (PCA)
  - Projection length behaves very well (better than $\Omega(1)$ error) when averaged over all data, but not guaranteed on each piece.
  - Theoretical use mainly in stochastic/mixture models. But wide practical use. Coming Soon.

Use of Dimension Reduction

- High Dimensional Nearest Neighbour Search (NNS):

Use of Dimension Reduction

- High Dimensional Nearest Neighbour Search (NNS):
  - Given a database of n points in $\mathbf{R}^d$. Preprocess in poly time.

Use of Dimension Reduction

- High Dimensional Nearest Neighbour Search (NNS):
  - Given a database of n points in $\mathbf{R}^d$. Preprocess in poly time.
  - Now, presented with a query point in $\mathbf{R}^d$, find (approximate) nearest database point to query point in **sub-linear** (i.e., $o(nd)$) time (or polylog time).

Use of Dimension Reduction

- High Dimensional Nearest Neighbour Search (NNS):
  - Given a database of n points in $\mathbf{R}^d$. Preprocess in poly time.
  - Now, presented with a query point in $\mathbf{R}^d$, find (approximate) nearest database point to query point in **sub-linear** (i.e., $o(nd)$) time (or polylog time).
  - Will see: projecting database points into a random polylog dim space is a good solution. At query time, will also project query point to same space and measure distances in the projection.

Use of Dimension Reduction

- High Dimensional Nearest Neighbour Search (NNS):
  - Given a database of n points in $\mathbf{R}^d$. Preprocess in poly time.
  - Now, presented with a query point in $\mathbf{R}^d$, find (approximate) nearest database point to query point in **sub-linear** (i.e., $o(nd)$) time (or polylog time).
  - Will see: projecting database points into a random polylog dim space is a good solution. At query time, will also project query point to same space and measure distances in the projection.
- (Over-determined) Linear Regression:

Use of Dimension Reduction

- High Dimensional Nearest Neighbour Search (NNS):
  - Given a database of n points in $\mathbf{R}^d$. Preprocess in poly time.
  - Now, presented with a query point in $\mathbf{R}^d$, find (approximate) nearest database point to query point in **sub-linear** (i.e., $o(nd)$) time (or polylog time).
  - Will see: projecting database points into a random polylog dim space is a good solution. At query time, will also project query point to same space and measure distances in the projection.
- (Over-determined) Linear Regression:
  - Given $n \times d$ ($n > d$) matrix A and n vector $\mathbf{b}$, find $\mathbf{x}$ minimizing $|A\mathbf{x} - \mathbf{b}|$.

Use of Dimension Reduction

- High Dimensional Nearest Neighbour Search (NNS):
  - Given a database of n points in $\mathbf{R}^d$. Preprocess in poly time.
  - Now, presented with a query point in $\mathbf{R}^d$, find (approximate) nearest database point to query point in **sub-linear** (i.e., $o(nd)$) time (or polylog time).
  - Will see: projecting database points into a random polylog dim space is a good solution. At query time, will also project query point to same space and measure distances in the projection.
- (Over-determined) Linear Regression:
  - Given $n \times d$ $(n > d)$ matrix A and n vector $\mathbf{b}$, find $\mathbf{x}$ minimizing $|A\mathbf{x} - \mathbf{b}|$.
  - Take a random poly$(d) \times n$ matrix P. Solve instead $|PA\mathbf{x} - P\mathbf{b}|$ (no n. Only d - gain if $d << n$)

Use of Dimension Reduction

- High Dimensional Nearest Neighbour Search (NNS):
  - Given a database of n points in $\mathbf{R}^d$. Preprocess in poly time.
  - Now, presented with a query point in $\mathbf{R}^d$, find (approximate) nearest database point to query point in **sub-linear** (i.e., $o(nd)$) time (or polylog time).
  - Will see: projecting database points into a random polylog dim space is a good solution. At query time, will also project query point to same space and measure distances in the projection.
- (Over-determined) Linear Regression:
  - Given $n \times d$ ($n > d$) matrix A and n vector $\mathbf{b}$, find $\mathbf{x}$ minimizing $|A\mathbf{x} - \mathbf{b}|$.
  - Take a random poly(d) $\times$ n matrix P. Solve instead $|PA\mathbf{x} - P\mathbf{b}|$ (no n. Only d - gain if $d << n$)
- Many other Examples : $k-$means Clustering. How about $k-$median ? [Discussion later.]

Length Preserving Projection

- Want a random "length preserving" linear projection from $\mathbf{R}^d \to \mathbf{R}^k$, $k << d$:

Length Preserving Projection

- Want a random "length preserving" linear projection from $\mathbf{R}^d \rightarrow \mathbf{R}^k$, $k << d$:
- I.e., want $k \times d$ random P so that $\forall \mathbf{v} \in \mathbf{R}^d$, $\Pr(|P\mathbf{v}| = (1 \pm \varepsilon)|\mathbf{v}|) \geq 1 - \delta$.

Length Preserving Projection

- Want a random "length preserving" linear projection from $\mathbf{R}^d \to \mathbf{R}^k$, $k << d$:
- I.e., want $k \times d$ random P so that $\forall \mathbf{v} \in \mathbf{R}^d$, $\Pr(|P\mathbf{v}| = (1 \pm \varepsilon)|\mathbf{v}|) \geq 1 - \delta$.
- Note the placement of $\forall$ quantifier. What happens if I place it inside $\Pr(..)$?

Length Preserving Projection

- Want a random "length preserving" linear projection from $\mathbf{R}^d \to \mathbf{R}^k$, $k << d$:
- I.e., want $k \times d$ random P so that $\forall \mathbf{v} \in \mathbf{R}^d$, $\Pr(|P\mathbf{v}| = (1 \pm \varepsilon)|\mathbf{v}|) \geq 1 - \delta$.
- Note the placement of $\forall$ quantifier. What happens if I place it inside $\Pr(..)$?
- Not true if quantifier is inside: since $k < d$, there is a non-zero $\mathbf{v}$ in null space of P...

Length Preserving Projection

- Want a random "length preserving" linear projection from $\mathbf{R}^d \to \mathbf{R}^k$, $k << d$:
- I.e., want $k \times d$ random P so that $\forall \mathbf{v} \in \mathbf{R}^d$, $\Pr(|P\mathbf{v}| = (1 \pm \varepsilon)|\mathbf{v}|) \geq 1 - \delta$.
- Note the placement of $\forall$ quantifier. What happens if I place it inside $\Pr(..)$?
- Not true if quantifier is inside: since $k < d$, there is a non-zero $\mathbf{v}$ in null space of P...
- Original projection: Project to a random $k$dimensional subspace. In terms of matrices, P is a random $k \times d$ matrix with orthonormal rows.

Length Preserving Projection

- Want a random "length preserving" linear projection from $\mathbf{R}^d \to \mathbf{R}^k$, $k << d$:
- I.e., want $k \times d$ random P so that $\forall \mathbf{v} \in \mathbf{R}^d$, $\Pr\left(|P\mathbf{v}| = (1 \pm \varepsilon)|\mathbf{v}|\right) \geq 1 - \delta$.
- Note the placement of $\forall$ quantifier. What happens if I place it inside $\Pr(..)$?
- Not true if quantifier is inside: since $k < d$, there is a non-zero $\mathbf{v}$ in null space of P...
- Original projection: Project to a random $k$dimensional subspace. In terms of matrices, P is a random $k \times d$ matrix with orthonormal rows.
- How does one pick such a random matrix ? Dependence. Also proof of length-preserving property is hard because of the orthonormal requirement.

Projection with Gaussian Vectors

- Pick k iid vectors $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_k}$, each $N(\mathbf{0}, I)$.

Projection with Gaussian Vectors

- Pick k iid vectors $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_k}$, each $N(\mathbf{0}, I)$.
  - Recall: Var-Cov matrix of random vector $\mathbf{u}$ is a $d \times d$ matrix with (i, j) th entry equal to $E((u_i - E(u_i))(u_j - E(u_j)))$. It equals I means independent coordinates.

Projection with Gaussian Vectors

- Pick k iid vectors $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_k}$, each $N(\mathbf{0}, I)$.
  - Recall: Var-Cov matrix of random vector $\mathbf{u}$ is a $d \times d$ matrix with (i, j) th entry equal to $E((u_i - E(u_i))(u_j - E(u_j)))$. It equals I means independent coordinates.
- Projection $f : \mathbf{R}^d \to \mathbf{R}^k$ is given by $f(\mathbf{x}) = (\mathbf{x} \cdot \mathbf{u_1}, \mathbf{x} \cdot \mathbf{u_2}, \ldots, \mathbf{x} \cdot \mathbf{u_k})$, i.e., dot products of $\mathbf{x}$ with the k random vectors.

Projection with Gaussian Vectors

- Pick k iid vectors $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_k}$, each $N(\mathbf{0}, I)$.
  - Recall: Var-Cov matrix of random vector $\mathbf{u}$ is a $d \times d$ matrix with (i,j) th entry equal to $E((u_i - E(u_i))(u_j - E(u_j)))$. It equals I means independent coordinates.
- Projection $f : \mathbf{R}^d \to \mathbf{R}^k$ is given by $f(\mathbf{x}) = (\mathbf{x} \cdot \mathbf{u_1}, \mathbf{x} \cdot \mathbf{u_2}, \ldots, \mathbf{x} \cdot \mathbf{u_k})$, i.e., dot products of $\mathbf{x}$ with the k random vectors.
- Will prove that for each $\mathbf{v}$, whp, we have $|f(\mathbf{v})| = (1 \pm \varepsilon)\sqrt{k}|\mathbf{v}|$. Why $\sqrt{k}$? What is $E((\mathbf{v} \cdot \mathbf{u_1})^2)$ OR at least an upper bound?

Projection with Gaussian Vectors

- Pick k iid vectors $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_k}$, each $N(\mathbf{0}, I)$.
  - Recall: Var-Cov matrix of random vector $\mathbf{u}$ is a $d \times d$ matrix with (i, j) th entry equal to $E((u_i - E(u_i))(u_j - E(u_j)))$. It equals I means independent coordinates.
- Projection $f : \mathbf{R}^d \to \mathbf{R}^k$ is given by $f(\mathbf{x}) = (\mathbf{x} \cdot \mathbf{u_1}, \mathbf{x} \cdot \mathbf{u_2}, \ldots, \mathbf{x} \cdot \mathbf{u_k})$, i.e., dot products of $\mathbf{x}$ with the k random vectors.
- Will prove that for each $\mathbf{v}$, whp, we have $|f(\mathbf{v})| = (1 \pm \varepsilon)\sqrt{k}|\mathbf{v}|$. Why $\sqrt{k}$? What is $E((\mathbf{v} \cdot \mathbf{u_1})^2)$ OR at least an upper bound?
- Upper bound: If $\mathbf{v} \cdot \mathbf{u_1} = \sum_{j=1}^{d}(v_j u_{1j})$ is the sum of d independent Gaussians; means and variances add up. So, $\mathbf{v} \cdot \mathbf{u_1} \sim N(0, |\mathbf{v}|^2)$; thus, whp, $|\mathbf{v} \cdot \mathbf{u_1}| \leq c|\mathbf{v}|$. Note that $c|\mathbf{v}| \approx c|\mathbf{v}|\ |\mathbf{u_1}|/\sqrt{d}$, so this is an "equator" like bound - why ?

Random Projection Theorem

Theorem 1 Let f be as above. There is a constant $c > 0$ such that for $\varepsilon \in (0, 1)$,

$$\forall \mathbf{v} \in \mathbf{R}^d : \Pr\underbrace{\left(\Big| |f(\mathbf{v})| \ - \ \sqrt{k}|\mathbf{v}| \Big| \ \geq \varepsilon\sqrt{k}|\mathbf{v}| \ \right)}_{|f(\mathbf{v})| \approx_\varepsilon \sqrt{k}|\mathbf{v}|} \leq 3e^{-ck\varepsilon^2},$$

where the probability is taken over the random draws of vectors $\mathbf{u_i}$ used to construct f.

Theorem 2 For any $0 < \varepsilon < 1$ and any integer n, let $k \geq \frac{3}{c\varepsilon^2} \ln n$ for c as in Theorem 1. Suppose $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}$ is any set of n points.
$\Pr\left(\forall i, j \in \{1, 2, \ldots, n\} \left|f(\mathbf{v_i}) - f(\mathbf{v_j})\right| \approx_\varepsilon \sqrt{k}\left|\mathbf{v_i} - \mathbf{v_j}\right|\right) \geq 1 - \frac{2}{n}$.

Remarks on the Theorems

- Note: The claim is for EVERY $\mathbf{v}$. What if I just wanted a weaker statement - for MOST $\mathbf{v}$? Is it simpler? Would the weaker statement - for MOST be enough if we want to solve the NNS problem when the input is a RANDOM set of points?

Remarks on the Theorems

- Note: The claim is for EVERY $\mathbf{v}$. What if I just wanted a weaker statement - for MOST $\mathbf{v}$? Is it simpler? Would the weaker statement - for MOST be enough if we want to solve the NNS problem when the input is a RANDOM set of points?
- Advantage of Linearity of f: Estimate distance between two points $\mathbf{v_1}, \mathbf{v_2} \in \mathbf{R}^d$ is whp $(1 \pm \varepsilon)\frac{1}{\sqrt{k}}$ times distance between $f(\mathbf{v_1})$ and $f(\mathbf{v_2})$, since $f(\mathbf{v_1} - \mathbf{v_2}) = f(\mathbf{v_1}) - f(\mathbf{v_2})$.

Remarks on the Theorems

- Note: The claim is for EVERY $\mathbf{v}$. What if I just wanted a weaker statement - for MOST $\mathbf{v}$? Is it simpler? Would the weaker statement - for MOST be enough if we want to solve the NNS problem when the input is a RANDOM set of points?
- Advantage of Linearity of f: Estimate distance between two points $\mathbf{v_1}, \mathbf{v_2} \in \mathbf{R}^d$ is whp $(1 \pm \varepsilon)\frac{1}{\sqrt{k}}$ times distance between $f(\mathbf{v_1})$ and $f(\mathbf{v_2})$, since $f(\mathbf{v_1} - \mathbf{v_2}) = f(\mathbf{v_1}) - f(\mathbf{v_2})$.
- k being in the exponent in Theorem 1 is crucial to get $k \in O(\ln n)$ in Theorem 2.

Remarks on the Theorems

- Note: The claim is for EVERY $\mathbf{v}$. What if I just wanted a weaker statement - for MOST $\mathbf{v}$? Is it simpler? Would the weaker statement - for MOST be enough if we want to solve the NNS problem when the input is a RANDOM set of points?
- Advantage of Linearity of f: Estimate distance between two points $\mathbf{v_1}, \mathbf{v_2} \in \mathbf{R}^d$ is whp $(1 \pm \varepsilon)\frac{1}{\sqrt{k}}$ times distance between $f(\mathbf{v_1})$ and $f(\mathbf{v_2})$, since $f(\mathbf{v_1} - \mathbf{v_2}) = f(\mathbf{v_1}) - f(\mathbf{v_2})$.
- k being in the exponent in Theorem 1 is crucial to get $k \in O(\ln n)$ in Theorem 2.
- Many other random projections are now known. For example, the $\mathbf{u_i}$ can be taken as ± 1 vectors. Intuitively, if d is large, then $\mathbf{v} \cdot \mathbf{u_i}$ behaves as if it is a Gaussian r.v. But for small d, we need more care to argue this.

Proof of Theorem 1

- Want to prove: $|f(\mathbf{v})| \approx_\varepsilon \sqrt{k}|\mathbf{v}|$. Scale both sides and assume $|\mathbf{v}| = 1$.

Proof of Theorem 1

- Want to prove: $|f(\mathbf{v})| \approx_\varepsilon \sqrt{k}|\mathbf{v}|$. Scale both sides and assume $|\mathbf{v}| = 1$.
- $\mathbf{u_i} \cdot \mathbf{v} = \sum_{j=1}^{d} u_{ij} v_j =$ sum of d independent r.v.'s $u_{ij} v_j$ distributed $N(0, v_j^2)$ respectively.

Proof of Theorem 1

- Want to prove: $|f(\mathbf{v})| \approx_\varepsilon \sqrt{k}|\mathbf{v}|$. Scale both sides and assume $|\mathbf{v}| = 1$.
- $\mathbf{u_i} \cdot \mathbf{v} = \sum_{j=1}^{d} u_{ij} v_j =$ sum of d independent r.v.'s $u_{ij} v_j$ distributed $N(0, v_j^2)$ respectively.
- Sum of independent Gaussian r.v.s' - means and variances just add up. So, $\mathbf{u_i} \cdot \mathbf{v} \sim N(0, \sum_j v_j^2) \equiv N(0, 1)$.

Proof of Theorem 1

- Want to prove: $|f(\mathbf{v})| \approx_\varepsilon \sqrt{k}|\mathbf{v}|$. Scale both sides and assume $|\mathbf{v}| = 1$.
- $\mathbf{u_i} \cdot \mathbf{v} = \sum_{j=1}^{d} u_{ij} v_j =$ sum of d independent r.v.'s $u_{ij} v_j$ distributed $N(0, v_j^2)$ respectively.
- Sum of independent Gaussian r.v.s' - means and variances just add up. So, $\mathbf{u_i} \cdot \mathbf{v} \sim N(0, \sum_j v_j^2) \equiv N(0, 1)$.
- $\mathbf{u_1} \cdot \mathbf{v}, \mathbf{u_2} \cdot \mathbf{v}, \ldots, \mathbf{u_k} \cdot \mathbf{v}$ are independent. $f(\mathbf{v}) \sim N(\mathbf{0}, I_{k \times k})$. Apply Gaussian Annulus Theorem:

Proof of Theorem 1

- Want to prove: $|f(\mathbf{v})| \approx_\varepsilon \sqrt{k}|\mathbf{v}|$. Scale both sides and assume $|\mathbf{v}| = 1$.
- $\mathbf{u_i} \cdot \mathbf{v} = \sum_{j=1}^{d} u_{ij} v_j$ = sum of d independent r.v.'s $u_{ij} v_j$ distributed $N(0, v_j^2)$ respectively.
- Sum of independent Gaussian r.v.s' - means and variances just add up. So, $\mathbf{u_i} \cdot \mathbf{v} \sim N(0, \sum_j v_j^2) \equiv N(0, 1)$.
- $\mathbf{u_1} \cdot \mathbf{v}, \mathbf{u_2} \cdot \mathbf{v}, \ldots, \mathbf{u_k} \cdot \mathbf{v}$ are independent. $f(\mathbf{v}) \sim N(\mathbf{0}, I_{k\times k})$. Apply Gaussian Annulus Theorem:
- Let $\beta = \varepsilon\sqrt{k}$. $\Pr\left(|f(\mathbf{v})| \approx_\varepsilon \sqrt{k}\right) =$
  $\Pr\left(|f(\mathbf{v})| \in [\sqrt{k} - \beta, \sqrt{k} + \beta]\right) \geq 1 - e^{-c\beta^2} = 1 - e^{-ck\varepsilon^2}$.

Theorem 2

Union Bound: $O(n^2)$ pairs. Prob of failure for each is at most $e^{-ck\varepsilon^2}$.
So with $k \in \Omega(\ln n/\varepsilon^2)$, the failure probability is driven down to $< 1/n^2$...
Very Important: Exponential in k failure prob means we need k to grow only logarithmically.

$k-$means Clustering

- The Problem: Given n points in $\mathbf{R}^d$, k, partition into k clusters to minimize the sum of squared distances of points to nearest cluster center.

$k-$means Clustering

- The Problem: Given n points in $\mathbf{R}^d$, k, partition into k clusters to minimize the sum of squared distances of points to nearest cluster center.
- Cluster center=mean of cluster. [Only for $k-$means!]

$k-$means Clustering

- The Problem: Given n points in $\mathbf{R}^d$, k, partition into k clusters to minimize the sum of squared distances of points to nearest cluster center.
- Cluster center=mean of cluster. [Only for $k-$means!]
- ?? Obvious: Project to $O(\ln n/\varepsilon^2)$ dimensional subspace; find best clustering in projection. Will do? Since all distances are preserved to $1 \pm \varepsilon$.

$k-$means Clustering

- The Problem: Given n points in $\mathbf{R}^d$, k, partition into k clusters to minimize the sum of squared distances of points to nearest cluster center.
- Cluster center=mean of cluster. [Only for $k-$means!]
- ?? Obvious: Project to $O(\ln n/\varepsilon^2)$ dimensional subspace; find best clustering in projection. Will do? Since all distances are preserved to $1 \pm \varepsilon$.
- How Many distances need to be preserved? Crudely: How many possible cluster centers could there be ? 2^n since every subset of n points may form a cluster. Bad.

$k-$means Clustering

- The Problem: Given n points in $\mathbf{R}^d$, k, partition into k clusters to minimize the sum of squared distances of points to nearest cluster center.
- Cluster center=mean of cluster. [Only for $k-$means!]
- ?? Obvious: Project to $O(\ln n/\varepsilon^2)$ dimensional subspace; find best clustering in projection. Will do? Since all distances are preserved to $1 \pm \varepsilon$.
- How Many distances need to be preserved? Crudely: How many possible cluster centers could there be ? 2^n since every subset of n points may form a cluster. Bad.
- Luckily: For any m points: Sum of squared distances to the mean is $\frac{1}{|S|}$ times the sum of all pairwise distances among S. Useful?

$k-$means Clustering

- The Problem: Given n points in $\mathbf{R}^d$, k, partition into k clusters to minimize the sum of squared distances of points to nearest cluster center.
- Cluster center=mean of cluster. [Only for $k-$means!]
- ?? Obvious: Project to $O(\ln n/\varepsilon^2)$ dimensional subspace; find best clustering in projection. Will do? Since all distances are preserved to $1 \pm \varepsilon$.
- How Many distances need to be preserved? Crudely: How many possible cluster centers could there be ? 2^n since every subset of n points may form a cluster. Bad.
- Luckily: For any m points: Sum of squared distances to the mean is $\frac{1}{|S|}$ times the sum of all pairwise distances among S. Useful?
- Only need to preserve $O(n^2)$ pairwise distances, not 2^n.

$k-$means Clustering

- The Problem: Given n points in $\mathbf{R}^d$, k, partition into k clusters to minimize the sum of squared distances of points to nearest cluster center.
- Cluster center=mean of cluster. [Only for $k-$means!]
- ?? Obvious: Project to $O(\ln n/\varepsilon^2)$ dimensional subspace; find best clustering in projection. Will do? Since all distances are preserved to $1 \pm \varepsilon$.
- How Many distances need to be preserved? Crudely: How many possible cluster centers could there be ? 2^n since every subset of n points may form a cluster. Bad.
- Luckily: For any m points: Sum of squared distances to the mean is $\frac{1}{|S|}$ times the sum of all pairwise distances among S. Useful?
- Only need to preserve $O(n^2)$ pairwise distances, not 2^n.
- How about $k-$median clustering: Minimize sum of distances to cluster centers?

Gaussian Mixtures

- Gaussian Mixture is a probability density which is a convex combination of Gaussians.

Gaussian Mixtures

- Gaussian Mixture is a probability density which is a convex combination of Gaussians.
- For example, a mixture of k standard spherical Gaussians: $p(\mathbf{x}) = \sum_{t=1}^{k} w_t \frac{1}{(2\pi)^{d/2}} \exp\left(-|\mathbf{x} - \mu_{\mathbf{t}}|^2/2\right)$. $w_t \geq 0; \sum = 1$.

Gaussian Mixtures

- Gaussian Mixture is a probability density which is a convex combination of Gaussians.
- For example, a mixture of k standard spherical Gaussians: $p(\mathbf{x}) = \sum_{t=1}^{k} w_t \frac{1}{(2\pi)^{d/2}} \exp\left(-|\mathbf{x} - \mu_\mathbf{t}|^2/2\right)$. $w_t \geq 0; \sum = 1$.
- Samples from the mixture: $\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_n}$ i.i..d., each drawn according to $p(\mathbf{x})$.

Gaussian Mixtures

- Gaussian Mixture is a probability density which is a convex combination of Gaussians.
- For example, a mixture of k standard spherical Gaussians: $p(\mathbf{x}) = \sum_{t=1}^{k} w_t \frac{1}{(2\pi)^{d/2}} \exp\left(-|\mathbf{x} - \mu_\mathbf{t}|^2/2\right)$. $w_t \geq 0; \sum = 1$.
- Samples from the mixture: $\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_n}$ i.i..d., each drawn according to $p(\mathbf{x})$.
- Equivalent: For $i = 1, 2, \ldots, n$: Pick $t \in [k]$ acc to prob.s $w_1, w_2, \ldots, w_k$. Then, pick $\mathbf{x_i} \sim \frac{1}{(2\pi)^{d/2}} \exp\left(-|\mathbf{x} - \mu_\mathbf{t}|^2/2\right)$.

Gaussian Mixtures

- Gaussian Mixture is a probability density which is a convex combination of Gaussians.
- For example, a mixture of k standard spherical Gaussians: $p(\mathbf{x}) = \sum_{t=1}^{k} w_t \frac{1}{(2\pi)^{d/2}} \exp\left(-|\mathbf{x} - \mu_\mathbf{t}|^2/2\right)$. $w_t \geq 0; \sum = 1$.
- Samples from the mixture: $\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_n}$ i.i..d., each drawn according to $p(\mathbf{x})$.
- Equivalent: For $i = 1, 2, \ldots, n$: Pick $t \in [k]$ acc to prob.s $w_1, w_2, \ldots, w_k$. Then, pick $\mathbf{x_i} \sim \frac{1}{(2\pi)^{d/2}} \exp\left(-|\mathbf{x} - \mu_\mathbf{t}|^2/2\right)$.
- **Learning Problem** Given only samples $\mathbf{x_i}$, $i = 1, 2, \ldots, n$, find the t for each i. Clustering problem.

Gaussian Mixtures

- Gaussian Mixture is a probability density which is a convex combination of Gaussians.
- For example, a mixture of k standard spherical Gaussians: $p(\mathbf{x}) = \sum_{t=1}^{k} w_t \frac{1}{(2\pi)^{d/2}} \exp\left(-|\mathbf{x} - \mu_{\mathbf{t}}|^2/2\right)$. $w_t \geq 0; \sum = 1$.
- Samples from the mixture: $\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_n}$ i.i..d., each drawn according to $p(\mathbf{x})$.
- Equivalent: For $i = 1, 2, \ldots, n$: Pick $t \in [k]$ acc to prob.s $w_1, w_2, \ldots, w_k$. Then, pick $\mathbf{x_i} \sim \frac{1}{(2\pi)^{d/2}} \exp\left(-|\mathbf{x} - \mu_{\mathbf{t}}|^2/2\right)$.
- **Learning Problem** Given only samples $\mathbf{x_i}, i = 1, 2, \ldots, n$, find the t for each i. Clustering problem.
- Desired Solution: For $k \in O(1)$, want: "Can solve learning problem if the means of each pair of Gaussians are separated by $\Omega(1)$ (standard deviations) (which is 1 in here).

Gaussian Mixtures

- Gaussian Mixture is a probability density which is a convex combination of Gaussians.
- For example, a mixture of k standard spherical Gaussians: $p(\mathbf{x}) = \sum_{t=1}^{k} w_t \frac{1}{(2\pi)^{d/2}} \exp\left(-|\mathbf{x} - \mu_{\mathbf{t}}|^2/2\right)$. $w_t \geq 0; \sum = 1$.
- Samples from the mixture: $\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_n}$ i.i..d., each drawn according to $p(\mathbf{x})$.
- Equivalent: For $i = 1, 2, \ldots, n$: Pick $t \in [k]$ acc to prob.s $w_1, w_2, \ldots, w_k$. Then, pick $\mathbf{x_i} \sim \frac{1}{(2\pi)^{d/2}} \exp\left(-|\mathbf{x} - \mu_{\mathbf{t}}|^2/2\right)$.
- **Learning Problem** Given only samples $\mathbf{x_i}, i = 1, 2, \ldots, n$, find the t for each i. Clustering problem.
- Desired Solution: For $k \in O(1)$, want: "Can solve learning problem if the means of each pair of Gaussians are separated by $\Omega(1)$ (standard deviations) (which is 1 in here).
- Will see that "distance-based" clustering can do this if inter-mean separation is $\Omega(d^{1/4})$. Next chapter: SVD, can do with $\Omega(1)$ S.D.'s

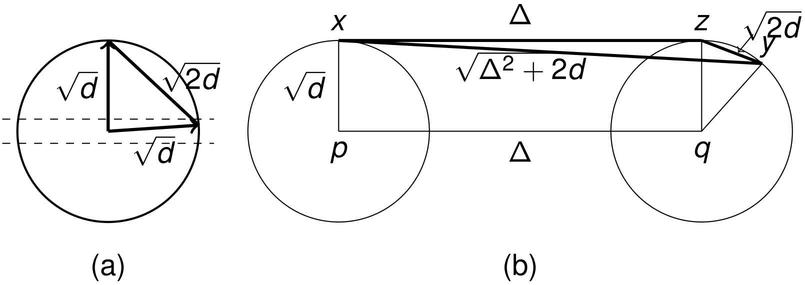

Figure: (a) indicates that two randomly chosen points in high dimension are surely almost nearly orthogonal. (b) indicates that the distance between a pair of random points from two different unit balls approximating the annuli of two Gaussians.

Two spherical Gaussians with unit variance (in every direction)

- If $\mathbf{x}, \mathbf{y}$ are two (indep) samples from the first Gaussian, then: $|\mathbf{x}-\mathbf{y}|^2 = |\mathbf{x}|^2 + |\mathbf{y}|^2 - 2\mathbf{x}\cdot\mathbf{y} =$ $(d \pm O(\sqrt{d})) \ + \ (d \pm O(\sqrt{d})) \pm O(\sqrt{d}) = 2d \pm O(\sqrt{d})$.

Two spherical Gaussians with unit variance (in every direction)

- If $\mathbf{x}, \mathbf{y}$ are two (indep) samples from the first Gaussian, then: $|\mathbf{x}-\mathbf{y}|^2 = |\mathbf{x}|^2 + |\mathbf{y}|^2 - 2\mathbf{x}\cdot\mathbf{y} =$ $(d \pm O(\sqrt{d})) \ + \ (d \pm O(\sqrt{d})) \pm O(\sqrt{d}) = 2d \pm O(\sqrt{d})$.
- If the two centers are Δ apart and $\mathbf{x}, \mathbf{z}$ are respectively from the two Gaussians, then

$$|\mathbf{x}-\mathbf{z}|^2 = |(\mathbf{x}-\mu_1) + (\mu_1-\mu_2) + (\mu_2-\mathbf{z})|^2 = d \pm O(\sqrt{d}) + \Delta^2 + d \pm \sqrt{d} +$$

Two spherical Gaussians with unit variance (in every direction)

- If $\mathbf{x}, \mathbf{y}$ are two (indep) samples from the first Gaussian, then: $|\mathbf{x}-\mathbf{y}|^2 = |\mathbf{x}|^2 + |\mathbf{y}|^2 - 2\mathbf{x}\cdot\mathbf{y} =$ $(d \pm O(\sqrt{d})) \ + \ (d \pm O(\sqrt{d})) \pm O(\sqrt{d}) = 2d \pm O(\sqrt{d})$.
- If the two centers are Δ apart and $\mathbf{x}, \mathbf{z}$ are respectively from the two Gaussians, then

$$|\mathbf{x}-\mathbf{z}|^2 = |(\mathbf{x}-\mu_1) + (\mu_1-\mu_2) + (\mu_2-\mathbf{z})|^2 = d \pm O(\sqrt{d}) + \Delta^2 + d \pm \sqrt{d} +$$

- Want: Whp, two points from the same Gaussian are closer to each other than 2 points from different Gaussians for "distance-based" clustering to succeed.

Two spherical Gaussians with unit variance (in every direction)

- If $\mathbf{x}, \mathbf{y}$ are two (indep) samples from the first Gaussian, then: $|\mathbf{x}-\mathbf{y}|^2 = |\mathbf{x}|^2 + |\mathbf{y}|^2 - 2\mathbf{x}\cdot\mathbf{y} =$ $(d \pm O(\sqrt{d})) \; + \; (d \pm O(\sqrt{d})) \pm O(\sqrt{d}) = 2d \pm O(\sqrt{d})$.
- If the two centers are Δ apart and $\mathbf{x}, \mathbf{z}$ are respectively from the two Gaussians, then

$$|\mathbf{x}-\mathbf{z}|^2 = |(\mathbf{x}-\mu_1) + (\mu_1-\mu_2) + (\mu_2-\mathbf{z})|^2 = d \pm O(\sqrt{d}) + \Delta^2 + d \pm \sqrt{d} +$$

- Want: Whp, two points from the same Gaussian are closer to each other than 2 points from different Gaussians for "distance-based" clustering to succeed.
- Suffices to have $\Delta^2 > c\sqrt{d}$ or $\Delta > cd^{1/4}$.

Two spherical Gaussians with unit variance (in every direction)

- If $\mathbf{x}, \mathbf{y}$ are two (indep) samples from the first Gaussian, then: $|\mathbf{x} - \mathbf{y}|^2 = |\mathbf{x}|^2 + |\mathbf{y}|^2 - 2\mathbf{x} \cdot \mathbf{y} =$ $(d \pm O(\sqrt{d})) \ + \ (d \pm O(\sqrt{d})) \pm O(\sqrt{d}) = 2d \pm O(\sqrt{d})$.
- If the two centers are Δ apart and $\mathbf{x}, \mathbf{z}$ are respectively from the two Gaussians, then

$$|\mathbf{x} - \mathbf{z}|^2 = |(\mathbf{x} - \mu_1) + (\mu_1 - \mu_2) + (\mu_2 - \mathbf{z})|^2 = d \pm O(\sqrt{d}) + \Delta^2 + d \pm \sqrt{d} +$$

- Want: Whp, two points from the same Gaussian are closer to each other than 2 points from different Gaussians for "distance-based" clustering to succeed.
- Suffices to have $\Delta^2 > c\sqrt{d}$ or $\Delta > cd^{1/4}$.
- If we want all pair of points to behave well (union bound) suffices to have $\Delta > cd^{1/4}\sqrt{\ln n}$.