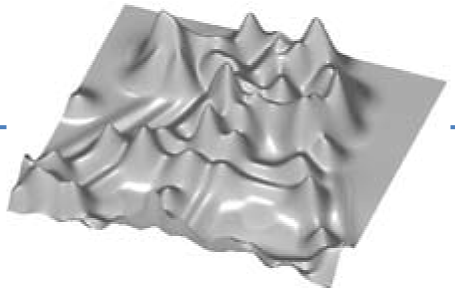
Randomized Algorithms

Sanjiv Kumar, Google Research, NY

EECS-6898, Columbia University - Fall, 2010

Curse of Dimensionality

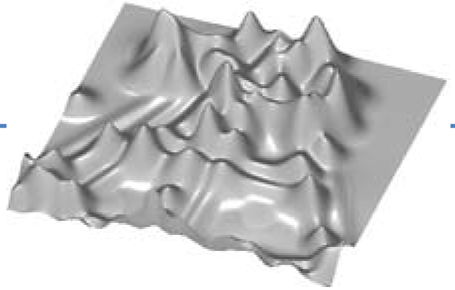


Gaussian Mixture Models (GMM)

- Density (likelihood) modeling
  - Can approximate any function arbitrarily close given enough components
- Clustering

$$p(x) = \sum_{j=1}^{m} P(j)p(x \mid j)$$

Curse of Dimensionality



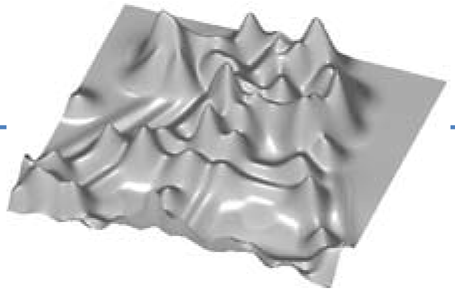
Gaussian Mixture Models (GMM)

- Density (likelihood) modeling
  - Can approximate any function arbitrarily close given enough components
- Clustering

$$p(x) = \sum_{j=1}^{m} P(j) p(x \mid j) = \sum_{j=1}^{m} \omega_j N(\mu_j, \Sigma_j) \longrightarrow d \times d$$

Mixing weights $\sum_{j=1}^{m} \omega_j = 1$

Curse of Dimensionality



Gaussian Mixture Models (GMM)

- Density (likelihood) modeling
  - Can approximate any function arbitrarily close given enough components
- Clustering

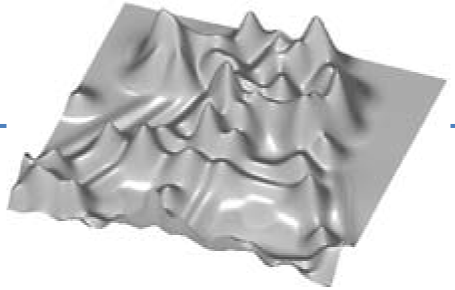
$$p(x) = \sum_{j=1}^{m} P(j)p(x \mid j) = \sum_{j=1}^{m} \omega_j N(\mu_j, \Sigma_j) \rightarrow d \times d$$

Mixing weights $\sum_{j=1}^{m} \omega_j = 1$

- Learning Via Expectation-Maximization
  - First-order method that iteratively fits a lower-bound on the data likelihood followed by maximization of the bound → commonly used with latent models
  - Learning:

Assignment probability $P_t(j \mid x_i) = \alpha\ p(x_i \mid j; \theta_t)\omega_j^t$

Curse of Dimensionality



Gaussian Mixture Models (GMM)

- Density (likelihood) modeling
  - Can approximate any function arbitrarily close given enough components
- Clustering

$$p(x) = \sum_{j=1}^{m} P(j)p(x \mid j) = \sum_{j=1}^{m} \omega_j N(\mu_j, \Sigma_j) \rightarrow d \times d$$

Mixing weights $\sum_{j=1}^{m} \omega_j = 1$

- Learning Via Expectation-Maximization
  - First-order method that iteratively fits a lower-bound on the data likelihood followed by maximization of the bound → commonly used with latent models
  - Learning

Assignment probability $\boxed{P_t(j \mid x_i) = \alpha\ p(x_i \mid j; \theta_t)\omega_j^t}$ $\quad \beta_j^t = \sum_i P_t(j \mid x_i; \theta_t)$

$\boxed{\omega_j^{t+1} = (1/n)\beta_j^t}$ $\boxed{\mu_j^{t+1} = (1/\beta_j^t)\sum_i p(j \mid x_i; \theta_t) x_i}$ $\boxed{\Sigma_j^{t+1} = (1/\beta_j^t)\sum_i p(j \mid x_i; \theta_t)(x_i - \mu_j^t)(x_i - \mu_j^t)^T}$

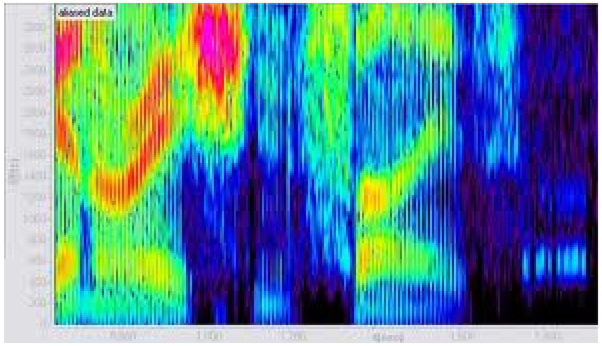
Curse of Dimensionality

Hidden Markov Models

- Time-series data
- Stock prices, Speech, Videos, Natural Language Processing

Video Categorization

Phoneme recognition



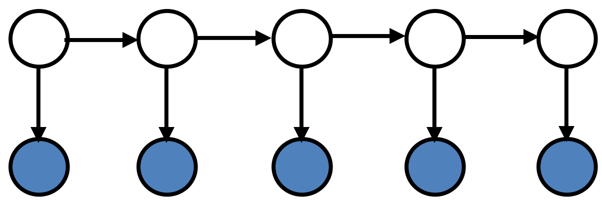
Curse of Dimensionality

Hidden Markov Models

- Time series data
- Stock prices, Speech, Videos, Natural Language Processing

For each category/ phoneme

state s_i



frame x_i

$$x = \{x_i\}, s = \{s_i\}$$

$$x_i \in \Re^d, s_i \in S$$

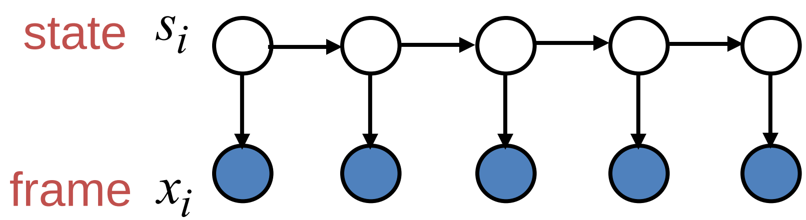
Evidence $p(x) = \sum_s p(x,s)$

Curse of Dimensionality

Hidden Markov Models

– Time series data
– Stock prices, Speech, Videos, Natural Language Processing

For each category/ phoneme



$$x = \{x_i\}, s = \{s_i\}$$
$$x_i \in \Re^d, s_i \in S$$

Evidence $p(x) = \sum_s p(x,s)$

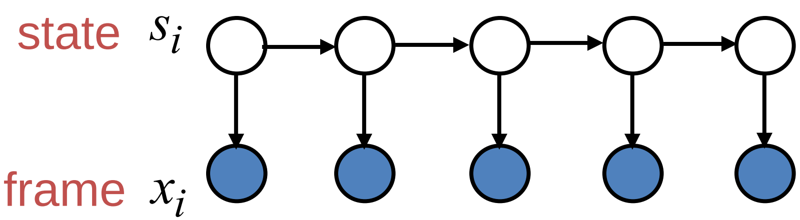
$$p(x,s) = p(x \mid s)\, p(s)$$
$$= P(s_1) \prod_i P(s_{i+1} \mid s_i) \prod_i p(x_i \mid s_i)$$

Curse of Dimensionality

Hidden Markov Models

- Time series data
- Stock prices, Speech, Videos, Natural Language Processing

For each category/ phoneme



$$x = \{x_i\}, s = \{s_i\}$$
$$x_i \in \Re^d, s_i \in S$$

Evidence $p(x) = \sum_s p(x,s)$

$$p(x,s) = p(x \mid s)p(s)$$
$$= P(s_1)\prod_i P(s_{i+1} \mid s_i)\prod_i p(x_i \mid s_i)$$

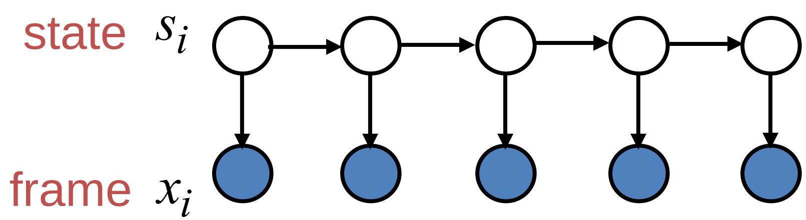
$$p(x_i \mid s_i = k) = \sum_{j=1}^{m} \omega_j^k N(\mu_j^k, \Sigma_j^k)$$

Curse of Dimensionality

Hidden Markov Models

- Time series data
- Stock prices, Speech, Videos, Natural Language Processing

For each category/ phoneme



$$x = \{x_i\}, s = \{s_i\}$$
$$x_i \in \Re^d, s_i \in S$$

Evidence $p(x) = \sum_s p(x,s)$

$$p(x,s) = p(x \mid s)\, p(s)$$
$$= P(s_1) \prod_i P(s_{i+1} \mid s_i) \prod_i p(x_i \mid s_i)$$

$$p(x_i \mid s_i = k) = \sum_{j=1}^{m} \omega_j^k N(\mu_j^k, \Sigma_j^k)$$

$d \times d$

full $O(10T)$

diagonal $O(100M)$

Video Analysis $d \sim O(100K)$ $|S| \sim O(100), m = O(10)$

Curse of Dimensionality

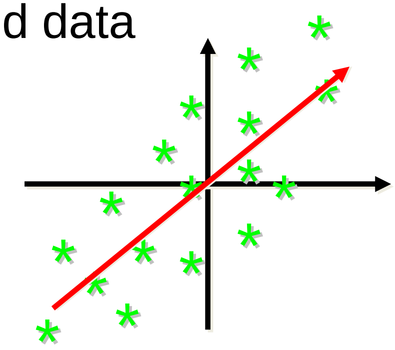
Nearest Neighbor Search

- Density estimation, classification, clustering/semi-supervised learning (graph-construction), …
- Brute Force: $O(nd)$
- If quality is not affected (much), can we reduce data to $d' << d$
- Ways of avoiding factor n discussed later

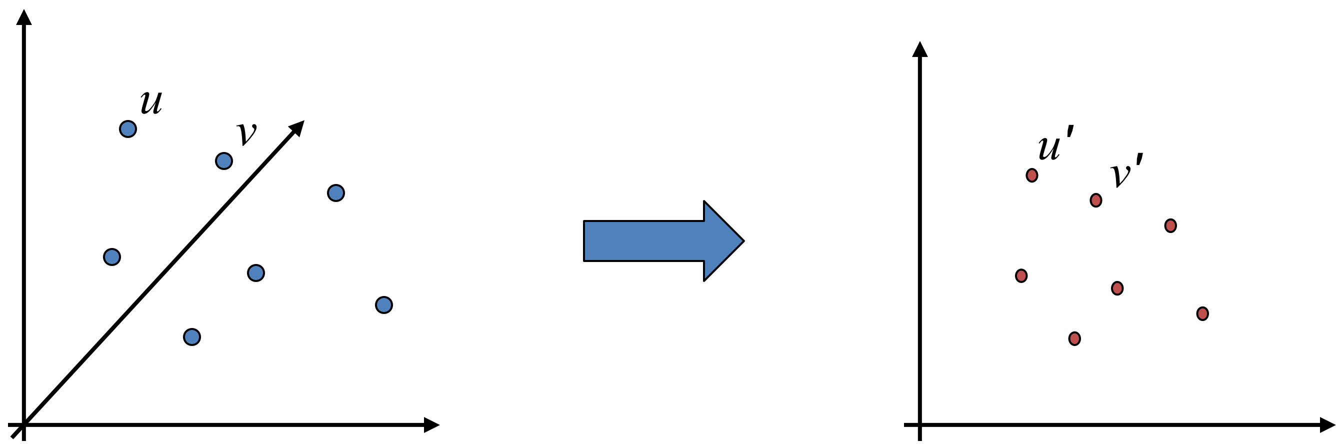
Data Visualization – hard to do in high-dimensional spaces

Dimensionality Reduction

- Linear methods e.g., PCA, metric MDS
- Finds directions that maximize variance of the projected data
- Also minimizes mean squared reconstruction error
- Computationally expensive $O(nd^2)$
- No worst case guarantees for distance preservation



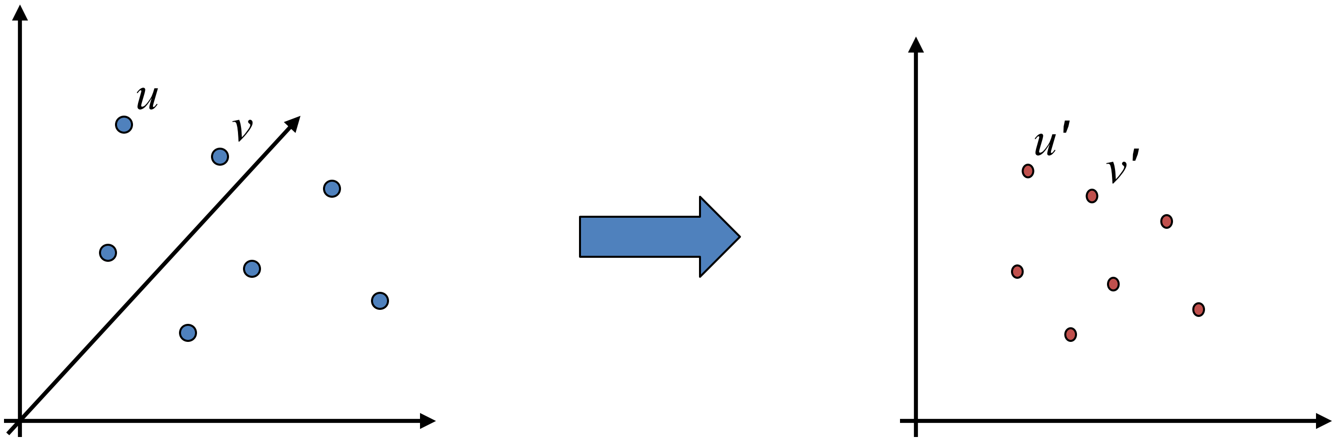
Linear Dimensionality Reduction



$$u' = R^T \; u$$

$$k \times 1 \quad k \times d \quad d \times 1 \qquad k \le d$$

Linear Dimensionality Reduction



$$u' = R^T \; u$$

$k \times 1 \quad k \times d \quad d \times 1 \qquad k \leq d$

Goal: To find an R such that $\|u' - v'\|^2 \approx \alpha \|u - v\|^2$

Does it exist?

Randomized Projections

Johnson-Lindenstrauss (JL) Lemma [1984]:

Given $0 < \varepsilon < 1$ and any integer n, let k be a positive integer such that,

$$k \geq 4(\varepsilon^2/2 - \varepsilon^3/3)^{-1} \log n = O(\varepsilon^{-2} \log n)$$

then, for any set P of n points in $\Re^d$, there exists a map $f : \Re^d \rightarrow \Re^k$ such that for all $u, v \in P$,

$$(1-\varepsilon)\|u-v\|_2^2 \leq \|f(u) - f(v)\|_2^2 \leq (1+\varepsilon)\|u-v\|_2^2$$

Randomized Projections

Johnson-Lindenstrauss (JL) Lemma [1984]:

Given $0 < \varepsilon < 1$ and any integer n, let k be a positive integer such that,

$$k \geq 4(\varepsilon^2/2 - \varepsilon^3/3)^{-1} \log n = O(\varepsilon^{-2} \log n)$$ no d !!

then, for any set P of n points in $\Re^d$, there exists a map $f : \Re^d \rightarrow \Re^k$ such that for all $u, v \in P$,

$$(1-\varepsilon)\|u-v\|_2^2 \leq \|f(u)-f(v)\|_2^2 \leq (1+\varepsilon)\|u-v\|_2^2$$

Example: Video Analysis

$$d = 100K \quad n = 100M \quad \varepsilon = 0.1$$

$$\boxed{k \approx 7K}$$

In practice, usually much smaller number (< 1K) is enough !

$$O(100M) \Rightarrow O(1M)$$

Construction of Mapping f

Consider linear mapping

$$f(u) = \sqrt{1/k}\; R^T u$$

Construction of Mapping f

Consider linear mapping

$$f(u) = \sqrt{1/k}\, R^T u$$

$d \times k$ Random Matrix

$r_{ij} \sim N(0,1)$

Key elements of the proof

- For any vector, squared length of its projection is sharply concentrated around its mean, i.e., squared length of original vector.

  (Hint: this is true even for difference of vectors: $(u - v)$)

- For a collection of n points, apply this observation to all pairs and try to bound maximum pairwise distortion to be within $(1 \pm \varepsilon)$ of mean

Elementary Proof

Single projection $q_j = r_j^T u$

We want $E(q_j^2)$

Expectation $E(q_j) = E(r_j^T)u = 0$ since $r_j \sim N(0, I)$

Variance $E(q_j^2) = u^T E(r_j r_j^T)u = \|u\|^2$

True for any distribution for which r_{ij} are iid, and $E(r_{ij}) = 0,\ E(r_{ij}^2) = 1$

Elementary Proof

Single projection $q_j = r_j^T u$

We want $E(q_j^2)$

Expectation $E(q_j) = E(r_j^T)u = 0$ since $r_j \sim N(0, I)$

Variance $E(q_j^2) = u^T E(r_j r_j^T)u = \|u\|^2$

True for any distribution for which r_{ij} are iid, and $E(r_{ij}) = 0,\ E(r_{ij}^2) = 1$

k-dim embedding $f(u) = \sqrt{1/k}\ R^T u$

$$\boxed{E(\|f(u)\|^2) = (1/k)\sum_j E(q_j^2) = \|u\|^2}$$

Next: To show that distribution of $\|f(u)\|^2$ is concentrated around mean

Elementary Proof

1-projection $q_j = r_j^T u \Rightarrow q_j \sim \|u\| N(0, 1) \equiv \|u\| x_j$

k-projections $f(u) = \sqrt{1/k}\, R^T u \Rightarrow \|f(u)\|^2 = \frac{1}{k}\sum_j q_j^2 = \frac{\|u\|^2}{k}\sum_j x_j^2$

Elementary Proof

1-projection $\quad q_j = r_j^T u \Rightarrow q_j \sim \|u\| N(0,1) \equiv \|u\| x_j$

k-projections $\quad f(u) = \sqrt{1/k}\; R^T u \Rightarrow \|f(u)\|^2 = \frac{1}{k}\sum_j q_j^2 = \frac{\|u\|^2}{k}\sum_j x_j^2 \quad X \sim \chi^2(k)$

$$\Pr[\|f(u)\|^2 \geq (1+\varepsilon)\|u\|^2] = \Pr[(1/k)\|u\|^2 X \geq (1+\varepsilon)\|u\|^2]$$

$$= \Pr[X \geq (1+\varepsilon)k]$$

Elementary Proof

1-projection $\quad q_j = r_j^T u \Rightarrow q_j \sim \|u\| N(0,1) \equiv \|u\| x_j$

k-projections $\quad f(u) = \sqrt{1/k}\, R^T u \Rightarrow \|f(u)\|^2 = \frac{1}{k}\sum_j q_j^2 = \frac{\|u\|^2}{k} \sum_j x_j^2 \quad X \sim \chi^2(k)$

$$\begin{aligned}
\Pr[\|f(u)\|^2 \geq (1+\varepsilon)\|u\|^2] &= \Pr[(1/k)\|u\|^2 X \geq (1+\varepsilon)\|u\|^2] \\
&= \Pr[X \geq (1+\varepsilon)k] \\
&= \Pr[e^{\lambda X} \geq e^{\lambda(1+\varepsilon)k}] \quad \forall\, \lambda \geq 0 \\
&\leq E[e^{\lambda X}] / e^{\lambda(1+\varepsilon)k}
\end{aligned}$$

Markov's inequality
$\Pr[|x| \geq a] \leq E[|x|]/a$

Elementary Proof

1-projection $\quad q_j = r_j^T u \;\Rightarrow\; q_j \sim \|u\| N(0,1) \equiv \|u\| x_j$

k-projections $\quad f(u) = \sqrt{1/k}\, R^T u \;\Rightarrow\; \|f(u)\|^2 = \frac{1}{k}\sum_j q_j^2 = \frac{\|u\|^2}{k} \sum_j x_j^2 \quad X \sim \chi^2(k)$

$$\Pr[\|f(u)\|^2 \geq (1+\varepsilon)\|u\|^2] = \Pr[(1/k)\|u\|^2 X \geq (1+\varepsilon)\|u\|^2]$$

$$= \Pr[X \geq (1+\varepsilon)k]$$

$$= \Pr[e^{\lambda X} \geq e^{\lambda(1+\varepsilon)k}] \quad \forall\, \lambda \geq 0$$

Markov's inequality
$\Pr[|x| \geq a] \leq E[|x|]/a$

$$\leq E[e^{\lambda X}]/e^{\lambda(1+\varepsilon)k}$$

mgf of $\chi^2(k)$

$$= (1/\sqrt{1-2\lambda}\, e^{\lambda(1+\varepsilon)})^k \quad \forall\, 1/2 \geq \lambda \geq 0$$

$\lambda = \varepsilon / 2(1+\varepsilon)$

$$= ((1+\varepsilon)e^{-\varepsilon})^{k/2}$$

Elementary Proof

1-projection $q_j = r_j^T u \Rightarrow q_j \sim \|u\| N(0,1) \equiv \|u\| x_j$

k-projections $f(u) = \sqrt{1/k}\, R^T u \Rightarrow \|f(u)\|^2 = \frac{1}{k}\sum_j q_j^2 = \frac{\|u\|^2}{k} \sum_j x_j^2$ $X \sim \chi^2(k)$

$$\Pr[\|f(u)\|^2 \geq (1+\varepsilon)\|u\|^2] = \Pr[(1/k)\|u\|^2 X \geq (1+\varepsilon)\|u\|^2]$$

$$= \Pr[X \geq (1+\varepsilon)k]$$

$$= \Pr[e^{\lambda X} \geq e^{\lambda(1+\varepsilon)k}] \quad \forall \lambda \geq 0$$

$$\leq E[e^{\lambda X}] / e^{\lambda(1+\varepsilon)k}$$

Markov's inequality $\Pr[|x| \geq a] \leq E[|x|]/a$

mgf of $\chi^2(k)$

$$= (1/\sqrt{1-2\lambda}\, e^{\lambda(1+\varepsilon)})^k \quad \forall 1/2 \geq \lambda \geq 0$$

$\lambda = \varepsilon / 2(1+\varepsilon)$

$$= ((1+\varepsilon)e^{-\varepsilon})^{k/2}$$

$\log(1+x) < x - x^2/2 + x^3/3$

$$\leq e^{-(\varepsilon^2/2 - \varepsilon^3/3)k/2}$$

$k = 4(\varepsilon^2/2 - \varepsilon^3/3)^{-1} \log n$

$$= n^{-2}$$

$$\boxed{\Pr[\|f(u)\|^2 \geq (1+\varepsilon)\|u\|^2] \leq 1/n^2}$$

Elementary Proof

Probabilities of large distortion

$$\boxed{\Pr[\|f(u)\|^2 \geq (1+\varepsilon)\|u\|^2] \leq 1/n^2}$$

$$\boxed{\Pr[\|f(u)\|^2 \leq (1-\varepsilon)\|u\|^2] \leq 1/n^2}$$

Elementary Proof

Probabilities of large distortion

$$\boxed{\Pr[\|f(u)\|^2 \geq (1+\varepsilon)\|u\|^2] \leq 1/n^2}$$

$$\boxed{\Pr[\|f(u)\|^2 \leq (1-\varepsilon)\|u\|^2] \leq 1/n^2}$$

Replacing u by $u - v$, and using linearity of f

large distortion probability for one pair of points $\leq 2/n^2$

l.d.p. for at least one among all pairs of points $\leq n(n-1)/2 \,.\, 2/n^2$

$= 1 - 1/n$

Elementary Proof

Probabilities of large distortion

$$\boxed{\Pr[\|f(u)\|^2 \geq (1+\varepsilon)\|u\|^2] \leq 1/n^2}$$

$$\boxed{\Pr[\|f(u)\|^2 \leq (1-\varepsilon)\|u\|^2] \leq 1/n^2}$$

Replacing u by $u-v$, and using linearity of f

large distortion probability for one pair of points $\leq 2/n^2$

l.d.p. for at least one among all pairs of points $\leq n(n-1)/2 \,.\, 2/n^2$

$= 1 - 1/n$

$$\boxed{\Pr[(1-\varepsilon)\|u-v\|^2 \leq \|f(u)-f(v)\|^2 \leq (1+\varepsilon)\|u-v\|^2] \leq 1/n}$$

Repeating this projection $O(n)$ times can boost the success probability to any desired constant.

Does this type of guarantee hold for L_1 also? unfortunately NO!

Application – Document retrieval

Given a database of n documents, retrieve similar to input document

Represent each document as a (tf-idf) bag-of-words feature vector

- Consider 2-grams of words as new “dimensions”: $d \sim O(1M)$

Application – Document retrieval

Given a database of n documents, retrieve similar to input document

Represent each document as a (tf-idf) bag-of-words feature vector

- Consider 2-grams of words as new "dimensions": $d \sim O(1M)$
- No problem! – JL: reduce to $k \sim 10\mathrm{K}$
- Generation of projection directions is fast – sample from std Gaussian

Application - Document retrieval

Given a database of n documents, retrieve similar to input document

Represent each document as a (tf-idf) bag-of-words feature vector

- Consider 2-grams of words as new "dimensions": $d \sim O(1M)$
- No problem! – JL: reduce to $k \sim 10\text{K}$
- Generation of projection directions is fast – sample from std Gaussian
- Two issues
  - Matrix multiplication: $O(ndk)$

$$n \sim O(1B)\ d \sim O(1M)\ k \sim O(10K) \Longrightarrow O(10MT)$$

Application - Document retrieval

Given a database of n documents, retrieve similar to input document

Represent each document as a (tf-idf) bag-of-words feature vector

- Consider 2-grams of words as new "dimensions": $d \sim O(1M)$
- No problem! – JL: reduce to $k \sim 10\text{K}$
- Generation of projection directions is fast – sample from std Gaussian
- Two issues
    - Matrix multiplication: $O(ndk)$

$$n \sim O(1B) \quad \underbrace{d \sim O(1M)}_{\text{usually sparse: } d_{nz} \sim O(1K)} \quad k \sim O(10K) \Rightarrow O(10MT)$$

usually sparse: $d_{nz} \sim O(1K)$ $\Rightarrow O(10KT)$

    - Memory requirement: $O(dk) \approx 40GB$!

Scale-friendly Random Projections

Instead of generating $r_{ij} \sim N(0,1)$, use a different distribution that has zero mean and unit variance

Scale-friendly Random Projections

Instead of generating $r_{ij} \sim N(0,1)$, use a different distribution that has zero mean and unit variance

$$r_{ij} = \begin{cases} +1 & \text{with } p = 1/2 \\ -1 & \text{with } p = 1/2 \end{cases}$$

$$r_{ij} = \sqrt{3} \begin{cases} +1 & \text{with } p = 1/6 \\ 0 & \text{with } p = 2/3 \\ -1 & \text{with } p = 1/6 \end{cases}$$

Scale-friendly Random Projections

Instead of generating $r_{ij} \sim N(0,1)$, use a different distribution that has zero mean and unit variance

	memory	*speed*
$r_{ij} = \begin{cases} +1 & \text{with } p = 1/2 \\ -1 & \text{with } p = 1/2 \end{cases}$	*1 bit/dim* $\approx 1.25GB$!	*Add half of the dims and subtract rest*
$r_{ij} = \sqrt{3}\begin{cases} +1 & \text{with } p = 1/6 \\ 0 & \text{with } p = 2/3 \\ -1 & \text{with } p = 1/6 \end{cases}$	*2 bit/dim* $\approx 2.5GB$!	*Add a few dims and subtract a few*

Scale-friendly Random Projections

Instead of generating $r_{ij} \sim N(0,1)$, use a different distribution that has zero mean and unit variance

	memory	speed
$r_{ij} = \begin{cases} +1 & \text{with } p = 1/2 \\ -1 & \text{with } p = 1/2 \end{cases}$	*1 bit/dim* $\approx 1.25GB$!	*Add half of the dims and subtract rest*
$r_{ij} = \sqrt{3}\begin{cases} +1 & \text{with } p = 1/6 \\ 0 & \text{with } p = 2/3 \\ -1 & \text{with } p = 1/6 \end{cases}$	*2 bit/dim* $\approx 2.5GB$!	*Add a few dims and subtract a few*

relation with "hash-kernel" ?

Similar guarantees as for $r_{ij} \sim N(0,1)$

Can we make the sampling matrix more sparse?

Fast JL Transform

Yes, one can make R more sparse but need to precondition the matrix to avoid excessive distortion

$$r_{ij} = \begin{cases} N(0,\ q^{-1}) & \text{with} \quad p = q \\ 0 & \text{with} \quad p = 1 - q \end{cases}$$

$$q = \min\{O(\varepsilon^{l-2} \log^{l} n/d), 1\}$$

$$\text{norm } l = \{1, 2\}$$

$$f(u) = R^T H D u$$

Fast JL Transform

Yes, one can make R more sparse but need to precondition the matrix to avoid excessive distortion

$$r_{ij} = \begin{cases} N(0,\ q^{-1}) & \text{with } p = q \\ 0 & \text{with } p = 1-q \end{cases}$$

$$q = \min\{O(\varepsilon^{l-2} \log^{l} n / d), 1\}$$

norm $l = \{1, 2\}$

$$f(u) = R^T H D u$$

$d \times d$

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

Diagonal Matrix

± 1 with $p = 1/2$

Fast JL Transform

Yes, one can make R more sparse but need to precondition the matrix to avoid excessive distortion

$$r_{ij} = \begin{cases} N(0,\ q^{-1}) & \text{with} \quad p = q \\ 0 & \text{with} \quad p = 1 - q \end{cases}$$

$$q = \min\{O(\varepsilon^{l-2} \log^{l} n/d), 1\}$$

$$\text{norm } l = \{1, 2\}$$

$$f(u) = R^T \underbrace{H}_{d \times d} \underbrace{D}_{d \times d} u$$

Randomized Fourier Transform

normalized Hadamard Matrix

orthogonal rows

$$d^{-1/2} \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

Diagonal Matrix

± 1 with $p = 1/2$

Fast JL Transform

Yes, one can make R more sparse but need to precondition the matrix to avoid excessive distortion

$$r_{ij} = \begin{cases} N(0,\ q^{-1}) & \text{with} \quad p = q \\ 0 & \text{with} \quad p = 1 - q \end{cases}$$

$$q = \min\{O(\varepsilon^{l-2} \log^l n/d), 1\}$$

norm $l = \{1, 2\}$

$$f(u) = \underbrace{R^T}_{k \times d} \underbrace{H}_{d \times d} \underbrace{D}_{d \times d} u$$

Randomized Fourier Transform

normalized Hadamard Matrix

orthogonal rows

$$d^{-1/2} \begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

Diagonal Matrix

± 1 with $p = 1/2$

Run Time: $O(d \log d) + ..$ with large constants

Random Projections in Practice ?

- Gaussian Mixture Models (GMMs) in high dimensions
- Classification
- Nearest Neighbor Search
  - Locality Sensitive Hashing (LSH)
  - Random Partitioning Trees
- Kernel Methods
  - Linearization of shift-invariant kernels
  - Reduction in computational complexity
    - Training: $O(n^3)$ to $O(n)$
    - Testing: $O(nd)$ to $O(k)$
- Matrix Approximations
  - Fast low-rank approximation
  - Accurate results for both dense and sparse matrices

Approximate Nearest Neighbors

Kernel Methods

Matrix Approximations

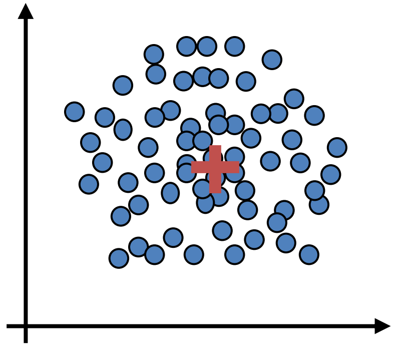
Application: High-Dim Mixture Models

Revisiting Video Analysis example: $d \sim O(100K)$

# of Gaussians: $m \sim O(1K)$

Weirdness of high-dimensional spaces:

$$p(x) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{1}{2\sigma^2}\|x-\mu\|^2\right)$$



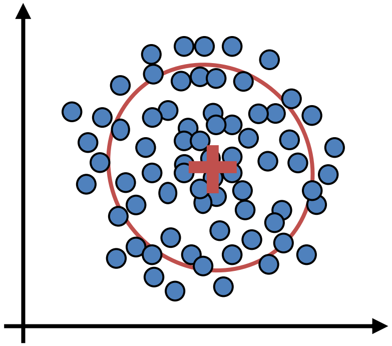
Application: High-Dim Mixture Models

Revisiting Video Analysis example: $d \sim O(100K)$

\# of Gaussians: $m \sim O(1K)$

Weirdness of high-dimensional spaces:

$$p(x) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{1}{2\sigma^2}\|x-\mu\|^2\right)$$



Randomly pick a point x from p(x)

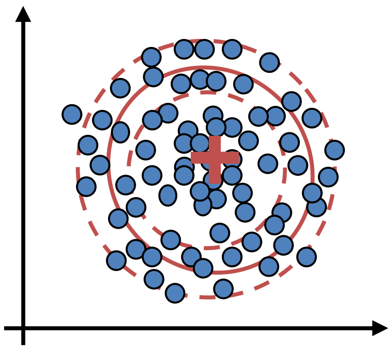
$$E[\|\mathrm{x}-\mu\|^2] = E[\sum_{i=1}^{d}(x_i-\mu_i)^2] = d\sigma^2$$

Application: High-Dim Mixture Models

Revisiting Video Analysis example: $d \sim O(100K)$

# of Gaussians: $m \sim O(1K)$

Weirdness of high-dimensional spaces:



$$p(x) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{1}{2\sigma^2} \|x - \mu\|^2 \right)$$

Randomly pick a point x from p(x)

$$E[\|\mathrm{x} - \mu\|^2] = E[\textstyle\sum_{i=1}^{d} (x_i - \mu_i)^2] = d\sigma^2$$

$$P[\left| \|\mathrm{x} - \mu\|^2 - d\sigma^2 \right| > \varepsilon\, d\sigma^2] \le 2e^{-d\varepsilon^2/24}$$

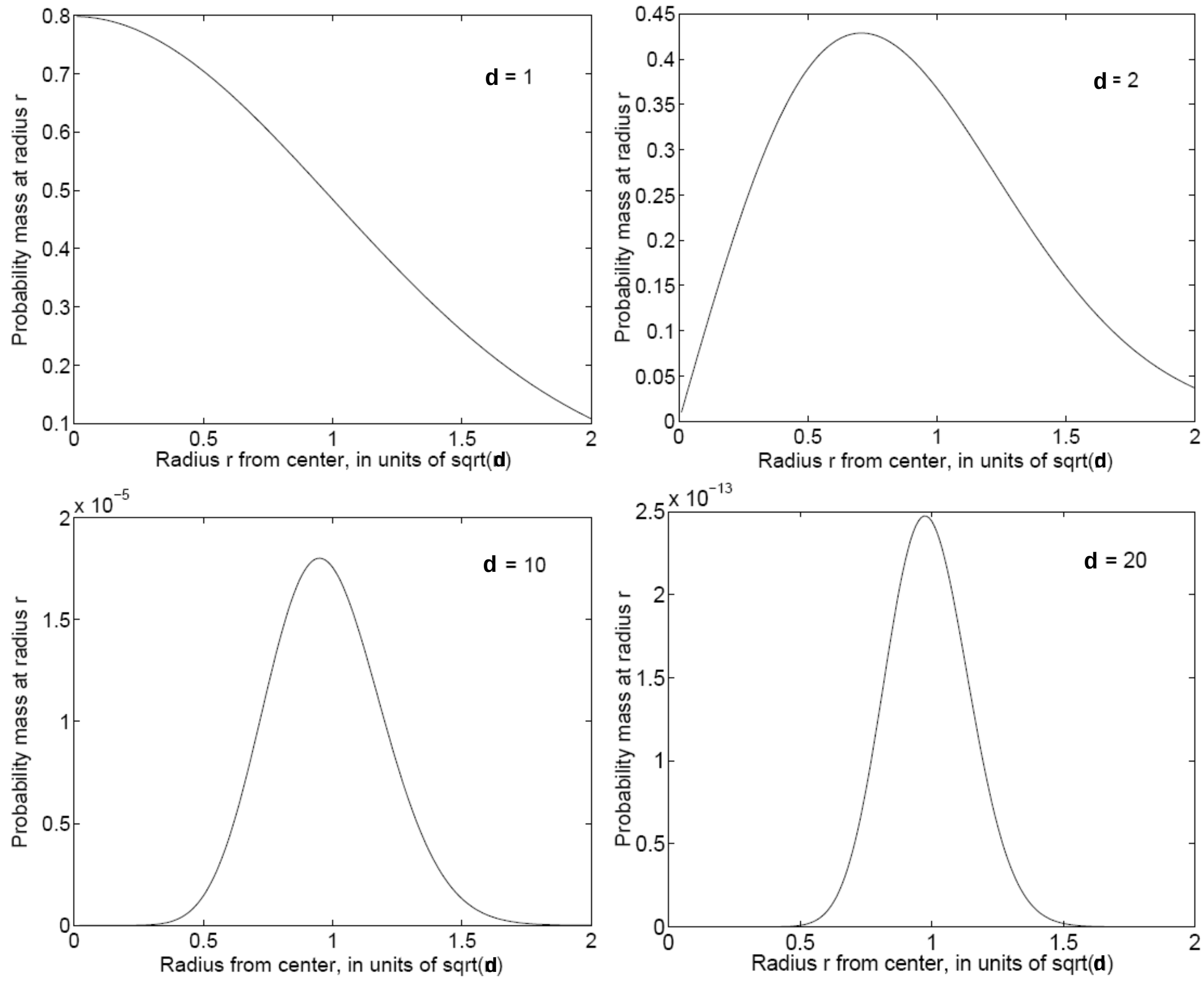
Although density is highest at μ, probability mass is concentrated in a thin shell around $\sigma\sqrt{d}$, i.e., we need O(2^d) points to learn reliably!!

Away from center, volume increases much more rapidly than the fall in density !

How about a uniform distribution in a hypercube?

data concentration in high-dim spaces

Single spherical gaussian with unit variance



Dasgupta [6]

Application: High-Dim Mixture Models

Project the data in low-dim space

1. Data can be projected in a very small subspace without significantly increasing the overlap of Gaussians

$$O(\varepsilon^{-2} \log m) \quad m \sim O(1K), \varepsilon = 0.1 \Rightarrow O(100)$$

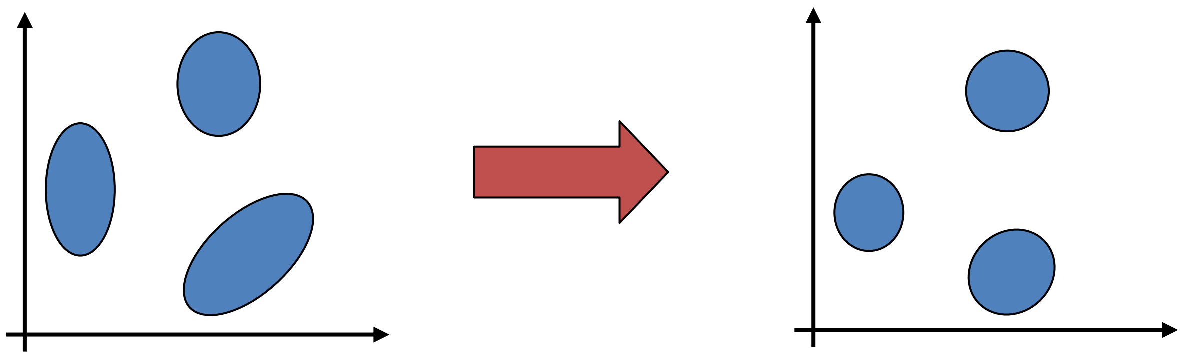
Independent of n and d !!

Application: High-Dim Mixture Models

Project the data in low-dim space

1. Data can be projected in a very small subspace without significantly increasing the overlap of Gaussians
$$O(\varepsilon^{-2} \log m) \quad m \sim O(1K), \varepsilon = 0.1 \Rightarrow O(100)$$
Independent of n and d !!

2. After projection, arbitrary ellipsoidal Gaussians become more spherical



Easier to learn parameters

Application: High-Dim Mixture Models

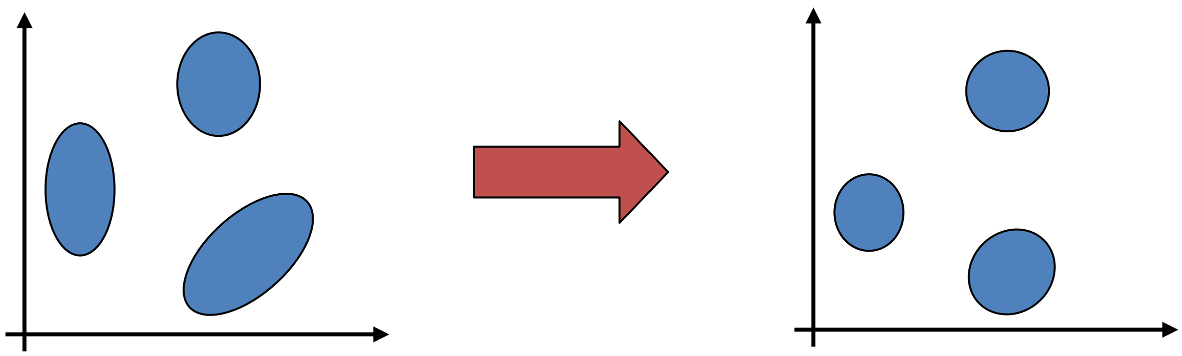
Project the data in low-dim space

1. Data can be projected in a very small subspace without significantly increasing the overlap of Gaussians

$$O(\varepsilon^{-2}\log m) \quad m \sim O(1K), \varepsilon = 0.1 \Rightarrow O(100)$$

Independent of n and d !!

2. After projection, arbitrary ellipsoidal Gaussians become more spherical



Easier to learn parameters

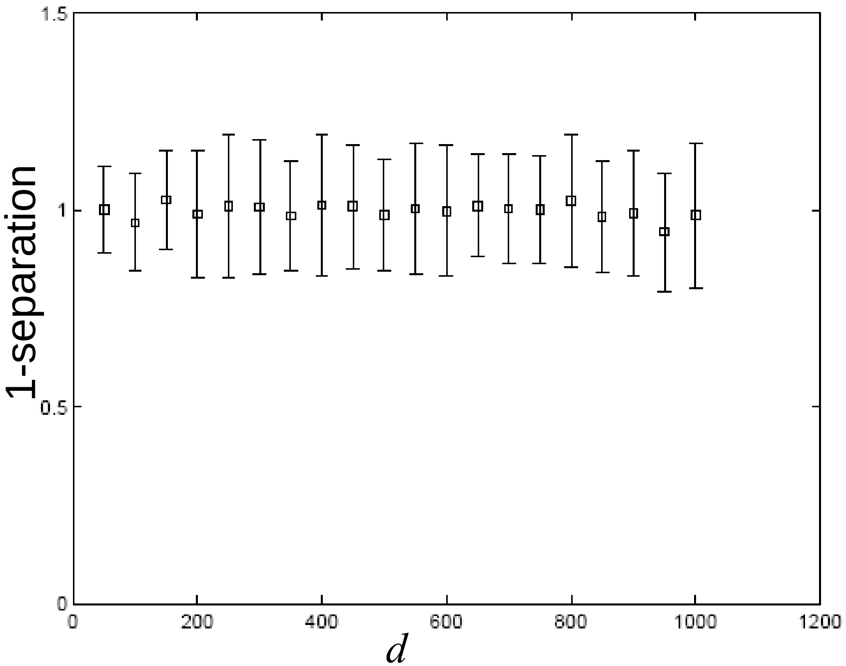
3. Mixture is learned in low-dim space and parameters of soft-clustering mapped back in original space.

Important in practice!

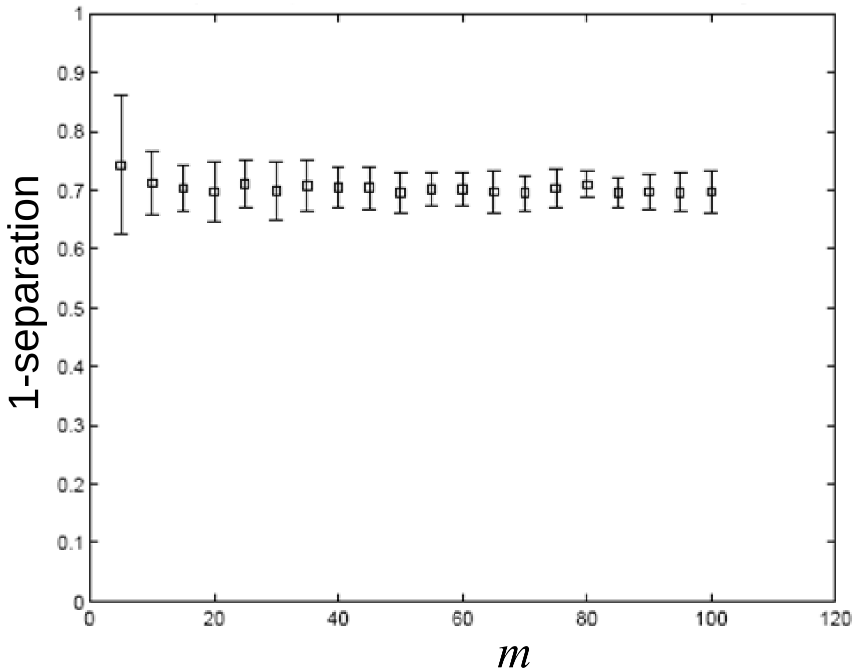
Examples - Overlap after projection

c-separation $\|\mu_1 - \mu_2\| \geq c\sqrt{\max\{tr(\Sigma_1), tr(\Sigma_2)\}}$

$$E\left(\|x-\mu\|^2\right) = tr(\Sigma)$$



$k = 20, m = 2$



$k = 10\log m$

Separation independent of d !

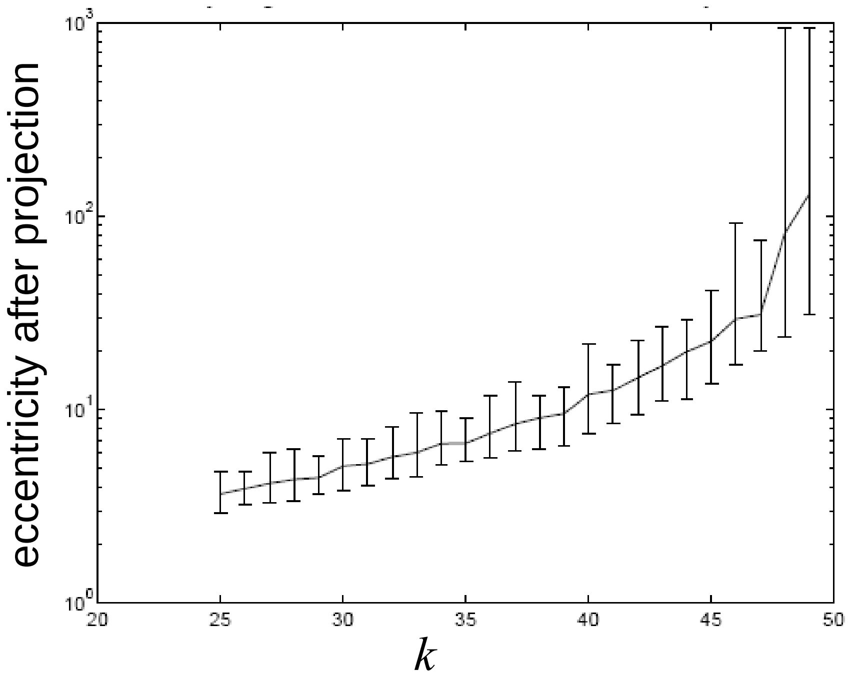
sufficient to maintain good separation

Dasgupta [6]

Examples - Eccentricity after projection

eccentricity $\lambda_{\max}(\Sigma) / \lambda_{\min}(\Sigma)$

Single Gaussian in 50-dim

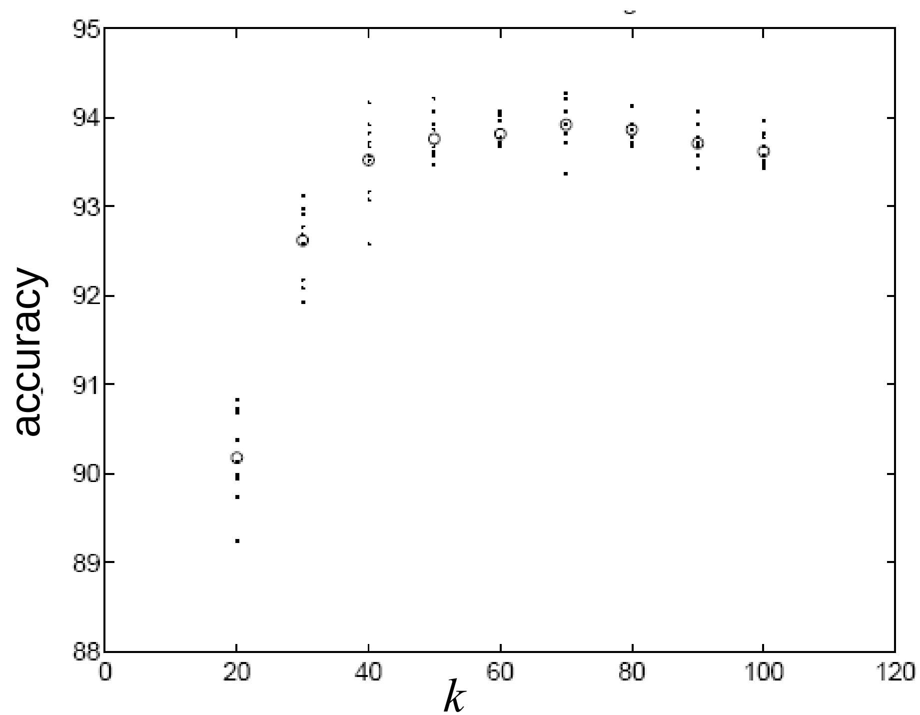


reducing k reduces eccentricity !

Dasgupta [6]

Examples - Accuracy

% accuracy on a handwritten digit classification set



$d = 256$ m = 5 per category, 10 categories

Dasgupta [6]

Random Projections Vs PCA – Classification

% accuracy on Ads dataset (UCI), n = 3279, d = 1554

Ads	C4.5		1NN		5NN		SVM	
	PCA	RP	PCA	RP	PCA	RP	PCA	RP
5	95.8 0.6	87.8 0.7	95.3 0.6	89.1 0.9	95.5 0.5	88.7 1.0	94.1 1.3	86.0 0.9
10	95.9 0.6	89.0 0.9	95.2 0.5	92.7 0.8	95.5 0.5	91.6 1.0	94.5 0.8	86.0 0.9
25	95.9 0.5	89.6 0.8	95.8 0.6	95.1 0.6	95.9 0.5	93.9 0.6	94.5 0.8	87.6 1.1
50	95.6 0.6	90.0 1.1	96.0 0.5	95.6 0.5	95.8 0.6	94.6 0.7	94.3 0.9	90.9 0.9
100	95.5 0.6	90.2 1.0	95.9 0.5	95.7 0.5	95.6 0.5	94.8 0.6	94.8 0.7	93.6 0.8
200	95.5 0.6	90.5 0.9	95.9 0.5	95.9 0.5	95.6 0.6	94.8 0.7	96.1 0.5	95.4 0.6
500	94.7 0.7	90.7 0.9	95.8 0.4	95.8 0.5	94.9 0.5	94.8 0.6	96.6 0.5	96.5 0.4
1554	96.0 0.6		95.8 0.5		94.7 0.6		96.8 0.4	

For low-dim projections, use PCA if computationally possible

Fradkin&Madigan [8]

Application: Kernel Linearization

Kernels commonly used for inducing nonlinearity

- Classification, regression, ranking, dimensionality reduction,…
- E.g., Kernel ridge regression, SVM, kernel logistic regression, kernel LDA, kernel PCA,…
- Powerful than their linear counterparts but higher computational costs
  - Training: $O(n^2)$-$O(n^3)$ Vs $O(n)$
  - Testing: $O(nd)$ Vs $O(d)$

Application: Kernel Linearization

Kernels commonly used for inducing nonlinearity

- Classification, regression, ranking, dimensionality reduction,…
- E.g., Kernel ridge regression, SVM, kernel logistic regression, kernel LDA, kernel PCA,…
- Powerful than their linear counterparts but higher computational costs
  - Training: $O(n^2)$-$O(n^3)$ Vs $O(n)$
  - Testing: $O(nd)$ Vs $O(d)$

Usually mercer kernels are used for inducing nonlinearity

$$k(x, y) = \Phi(x).\Phi(y)$$

Feature map for a generic kernel may not be known.

Application: Kernel Linearization

Kernels commonly used for inducing nonlinearity

- Classification, regression, ranking, dimensionality reduction,…
- E.g., Kernel ridge regression, SVM, kernel logistic regression, kernel LDA, kernel PCA,…
- Powerful than their linear counterparts but higher computational costs
  - Training: $O(n^2)$-$O(n^3)$ Vs $O(n)$
  - Testing: $O(nd)$ Vs $O(d)$

Usually mercer kernels are used for inducing nonlinearity

$$k(x,y) = \Phi(x).\Phi(y)$$

Feature map for a generic kernel may not be known.

Can we approximate the feature map with a low-dim vector ?

$$k(x,y) = \Phi(x).\Phi(y) \approx z(x).\underbrace{z(y)}_{\in \Re^{d'},\ d' << n}$$

Traditional approaches

Approaches to improve learning with kernels

- Decomposition methods (block coordinate-descent) → slow beyond O(100K) points
- Make the kernel matrix sparse by thresholding the entries
- Low-rank approximation of kernel matrix using column-sampling methods
- Hermite or Taylor approximation of kernel
- Approximate kernel matrix-vector product using ANN (kd-trees)

Instead of approximating the kernel matrix, directly approximate the feature map defining a kernel.

Traditional approaches

Approaches to improve learning with kernels

- Decomposition methods (block coordinate-descent) → slow beyond O(100K) points
- Make the kernel matrix sparse by thresholding the entries
- Low-rank approximation of kernel matrix using column-sampling methods
- Hermite or Taylor approximation of kernel
- Approximate kernel matrix-vector product using ANN (kd-trees)

Instead of approximating the kernel matrix, directly approximate the feature map defining a kernel.

Suppose the kernel is shift-invariant:

$$k(x, y) = k'(x - y) = k'(\Delta)$$

Gaussian

$$k(x, y) = \exp\{-\|x - y\|_2^2 / 2\sigma^2\}$$
$$k'(\Delta) = \exp\{-\|\Delta\|_2^2 / 2\sigma^2\}$$

Laplacian

$$k(x, y) = \exp\{-\|x - y\|_1 / \lambda\}$$
$$k'(\Delta) = \exp\{-\|\Delta\|_1 / \lambda\}$$

Random Fourier Features

Approximate $z(x) = [z_j(x)]_{d \times 1}$

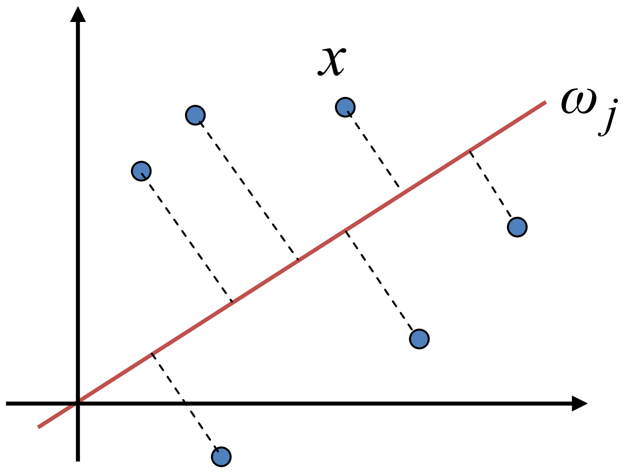
$$\boxed{z_j(x) = \cos(\omega_j x + b)} \quad \omega_j \sim P(\omega) \quad b \sim U(0, 2\pi)$$

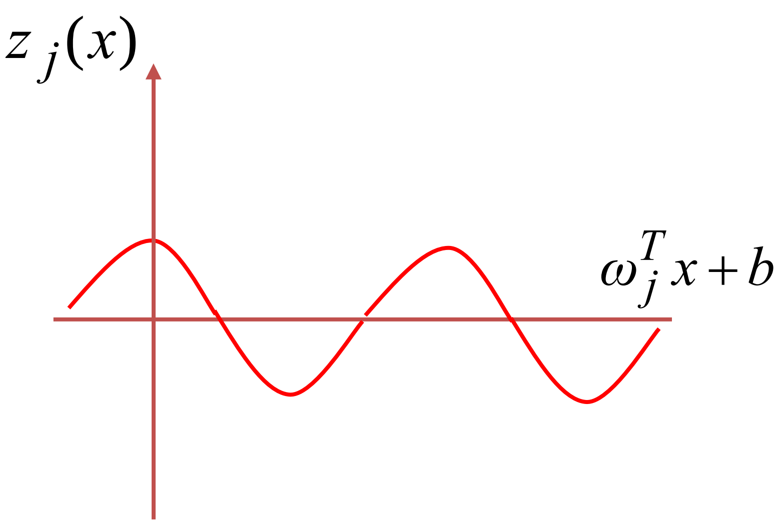
Random Fourier Features

Approximate $z(x) = [z_j(x)]_{d\times 1}$

$$\boxed{z_j(x) = \cos(\omega_j x + b)} \quad \omega_j \sim P(\omega) \quad b \sim U(0, 2\pi)$$

Gaussian $\omega_{jk} \sim N(0,1)$ Laplacian $\omega_{jk} \sim Cauchy(0,1)$





Main Theory

A continuous shift-invariant kernel is positive definite if and only if $k'(\Delta)$ is the Fourier transform of a non-negative measure. [Bochner]

$$k'(x-y) = \int p(\omega) e^{j\omega.(x-y)} d\omega$$

Main Theory

A continuous shift-invariant kernel is positive definite if and only if $k'(\Delta)$ is the Fourier transform of a non-negative measure. [Bochner]

$$k'(x-y) = \int p(\omega) e^{j\omega.(x-y)} d\omega$$

– since $k'(.)$ and $p(.)$ both are real, use real part of complex exponentials

$$k(x,y) = E\left[z_\omega(x) z_\omega(y)\right] \text{ if } z_\omega(x) = \sqrt{2}\cos(\omega^T x + b)$$

– Reduce variance by concatenating many (D) dimensions in $z_\omega(.)$

$$z_\omega(x)^T z_\omega(y) = (1/D)\sum_{j=1}^{D} z_{\omega_j}(x) z_{\omega_j}(y)$$

Main Theory

A continuous shift-invariant kernel is positive definite if and only if $k'(\Delta)$ is the Fourier transform of a non-negative measure. [Bochner]

$$k'(x-y) = \int p(\omega) e^{j\omega.(x-y)} d\omega$$

- since $k'(.)$ and $p(.)$ both are real, use real part of complex exponentials

$$k(x,y) = E\left[z_\omega(x) z_\omega(y)\right] \quad \text{if } z_\omega(x) = \sqrt{2}\cos(\omega^T x + b)$$

- Reduce variance by concatenating many (D) dimensions in $z_\omega(.)$

$$z_\omega(x)^T z_\omega(y) = (1/D)\sum_{j=1}^{D} z_{\omega_j}(x) z_{\omega_j}(y)$$

Hoeffding Bound $\quad \Pr\left(\left|z(x)^T z(y) - k(x,y)\right| \ge \varepsilon\right) \le 2\exp(-D\varepsilon^2/4)$

Example results

Regression and Classification errors

Training $\min_{w}\left(\left\|Z^T w - y\right\|_2^2 + \lambda\|w\|_2^2\right)$ Testing $f(x) = w^T z(x)$

Dataset	Fourier+LS	CVM	Exact SVM
CPU regression 6500 instances 21 dims	3.6% 20 secs $D = 300$	5.5% 51 secs	11% 31 secs ASVM
Census regression 18,000 instances 119 dims	5% 36 secs $D = 500$	8.8% 7.5 mins	9% 13 mins SVMTorch
Adult classification 32,000 instances 123 dims	14.9% 9 secs $D = 500$	14.8% 73 mins	15.1% 7 mins $\text{SVM}^{\text{light}}$
Forest Cover classification 522,000 instances 54 dims	11.6% 71 mins $D = 5000$	2.3% 7.5 hrs	2.2% 44 hrs libSVM
KDDCUP99 (see footnote) classification 4,900,000 instances 127 dims	7.3% 1.5 min $D = 50$	6.2% (18%) 1.4 secs (20 secs)	8.3% < 1 s SVM+sampling

References

1. Johnson & Lindenstrauss, “Extensions of Lipschitz mapping into a Hilbert space,” American Math Soc., 1984.
2. Indyk & Motwani, “Approximate nearest neighbors: towards removing the curse of dimensionality,” STOC, 1998.
3. Dasgupta, “Learning mixture of Gaussians,” FOCS, 1999.
4. Indyk, “Stable distributions, pseudorandom generators, embeddings and data stream computation”, FOCS, 2000.
5. Dasgupta, ”Experiments with random projection.” *UAI*, 2000.
6. Achlioptas, “Database-friendly random projections,” Symposium on Principles of Database Systems, 2001.
7. Brinkman and Charikar, “On the impossibility of dimension reduction in l1,” FOCS, 2003.
8. Fradkin and Madigan, “Experiments with Random Projections for Machine Learning,” KDD, 2003.
9. Ailon & Chazelle, “Approximate nearest neighbors and Fast Johnson-Lindenstrauss transform,” STOC, 2006.
10. Rahimi & Recht, “Random Features for Large-Scale Kernel Machines,” NIPS, 2007.
11. Halko, Martinsson, & Tropp, "Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions“, ACM Report, 2009.