Metric Embeddings

Ashley Montanaro

Department of Computer Science,
University of Bristol

December 12, 2008

Outline

- What is a metric and why would we want to embed one?
- Exponential dimensionality reduction
- Embedding finite metrics and applications

Metrics

A metric is a mathematical abstraction of the notion of distance.

Metrics

A metric is a mathematical abstraction of the notion of distance.

A metric space M is a pair (X, D), where X is a set of points and $D : X \times X \to [0, \infty)$ is a distance function satisfying:

Metrics

A metric is a mathematical abstraction of the notion of distance.

A metric space M is a pair (X, D), where X is a set of points and $D : X \times X \to [0, \infty)$ is a distance function satisfying:

- $D(p, q) = 0$ if and only if $p = q$

Metrics

A metric is a mathematical abstraction of the notion of distance.

A metric space M is a pair (X, D), where X is a set of points and $D : X \times X \to [0, \infty)$ is a distance function satisfying:

- $D(p, q) = 0$ if and only if $p = q$
- $D(p, q) = D(q, p)$

Metrics

A metric is a mathematical abstraction of the notion of distance.

A metric space M is a pair (X, D), where X is a set of points and $D : X \times X \to [0, \infty)$ is a distance function satisfying:

- $D(p, q) = 0$ if and only if $p = q$
- $D(p, q) = D(q, p)$
- Triangle inequality: $D(p, q) + D(q, r) \geq D(p, r)$

Metrics

A metric is a mathematical abstraction of the notion of distance.

A metric space M is a pair (X, D), where X is a set of points and $D : X \times X \to [0, \infty)$ is a distance function satisfying:

- $D(p, q) = 0$ if and only if $p = q$
- $D(p, q) = D(q, p)$
- Triangle inequality: $D(p, q) + D(q, r) \geq D(p, r)$

Example:

- String edit distance: $D(s, t)$ is the number of insertions, deletions and substitutions needed to change s into t

Finite metrics

Any metric on n points can be represented by a matrix M where $M_{ij} = D(i,j)$, e.g.:

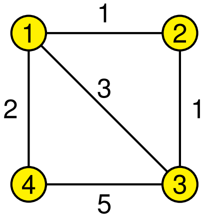
$$\begin{pmatrix} 0 & 1 & 3 & 2 \\ 1 & 0 & 4 & 1 \\ 3 & 4 & 0 & 3 \\ 2 & 1 & 3 & 0 \end{pmatrix}$$

Finite metrics

Any metric on n points can be represented by a matrix M where $M_{ij} = D(i,j)$, e.g.:

$$\begin{pmatrix} 0 & 1 & 3 & 2 \\ 1 & 0 & 4 & 1 \\ 3 & 4 & 0 & 3 \\ 2 & 1 & 3 & 0 \end{pmatrix}$$

A natural way of getting a finite metric is from the shortest path metric over a graph ($D(x,y) =$ length of shortest path from x to y).

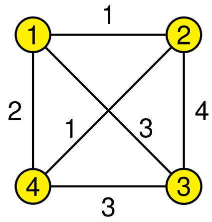


$$\longrightarrow \begin{pmatrix} 0 & 1 & 2 & 2 \\ 1 & 0 & 1 & 3 \\ 2 & 1 & 0 & 4 \\ 2 & 3 & 4 & 0 \end{pmatrix}$$

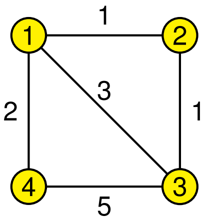
Finite metrics

Any metric on n points can be represented by a matrix M where $M_{ij} = D(i, j)$, e.g.:

$$\begin{pmatrix} 0 & 1 & 3 & 2 \\ 1 & 0 & 4 & 1 \\ 3 & 4 & 0 & 3 \\ 2 & 1 & 3 & 0 \end{pmatrix} \longrightarrow$$



A natural way of getting a finite metric is from the shortest path metric over a graph ($D(x, y) =$ length of shortest path from x to y).



$$\longrightarrow \begin{pmatrix} 0 & 1 & 2 & 2 \\ 1 & 0 & 1 & 3 \\ 2 & 1 & 0 & 4 \\ 2 & 3 & 4 & 0 \end{pmatrix}$$

Norms

A norm is a mathematical abstraction of the notion of length.

Norms

A norm is a mathematical abstraction of the notion of length.

A normed vector space is a pair $(V, \|\cdot\|)$, where V is a vector space and $\|\cdot\|$ is a norm, i.e. a function $V \to [0, \infty)$ satisfying:

Norms

A norm is a mathematical abstraction of the notion of length.

A normed vector space is a pair $(V, \|\cdot\|)$, where V is a vector space and $\|\cdot\|$ is a norm, i.e. a function $V \to [0, \infty)$ satisfying:

- $\|x\| = 0$ if and only if $x = 0$

Norms

A norm is a mathematical abstraction of the notion of length.

A normed vector space is a pair $(V, \|\cdot\|)$, where V is a vector space and $\|\cdot\|$ is a norm, i.e. a function $V \to [0, \infty)$ satisfying:

- $\|x\| = 0$ if and only if $x = 0$
- For $c \in \mathbb{R}$, $\|cx\| = |c|\|x\|$

Norms

A norm is a mathematical abstraction of the notion of length.

A normed vector space is a pair $(V, \|\cdot\|)$, where V is a vector space and $\|\cdot\|$ is a norm, i.e. a function $V \to [0, \infty)$ satisfying:

- $\|x\| = 0$ if and only if $x = 0$
- For $c \in \mathbb{R}$, $\|cx\| = |c|\|x\|$
- Triangle inequality: $\|x + y\| \leq \|x\| + \|y\|$

Norms give rise to metrics by setting $D(x, y) = \|x - y\|$.

Important norms

Some important examples are the ℓ_p norms: for $v \in \mathbb{R}^d$,

$$\|v\|_p = \left(\sum_{i=1}^{d} |v_i|^p \right)^{1/p}$$

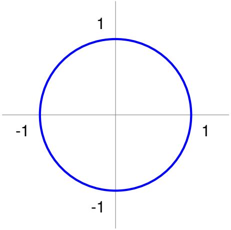
Important norms

Some important examples are the ℓ_p norms: for $v \in \mathbb{R}^d$,

$$\|v\|_p = \left(\sum_{i=1}^{d} |v_i|^p \right)^{1/p}$$

In particular:

- ℓ_2 (Euclidean) norm:
  $\|v\|_2 = \sqrt{\sum_i v_i^2}$



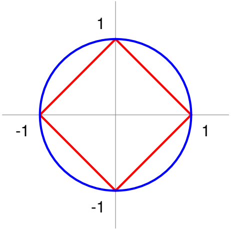
Important norms

Some important examples are the ℓ_p norms: for $v \in \mathbb{R}^d$,

$$\|v\|_p = \left(\sum_{i=1}^{d} |v_i|^p \right)^{1/p}$$

In particular:

- ℓ_1 (Manhattan/taxicab) norm: $\|v\|_1 = \sum_i |v_i|$
- ℓ_2 (Euclidean) norm: $\|v\|_2 = \sqrt{\sum_i v_i^2}$



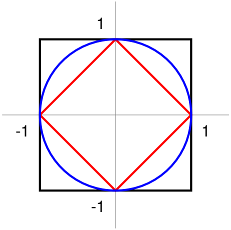
Important norms

Some important examples are the ℓ_p norms: for $v \in \mathbb{R}^d$,

$$\|v\|_p = \left(\sum_{i=1}^{d} |v_i|^p \right)^{1/p}$$

In particular:

- ℓ_1 (Manhattan/taxicab) norm: $\|v\|_1 = \sum_i |v_i|$
- ℓ_2 (Euclidean) norm: $\|v\|_2 = \sqrt{\sum_i v_i^2}$
- ℓ_∞ norm: $\|v\|_\infty = \max_i |v_i|$



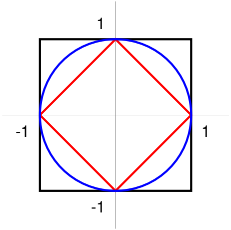
Important norms

Some important examples are the ℓ_p norms: for $v \in \mathbb{R}^d$,

$$\|v\|_p = \left(\sum_{i=1}^{d} |v_i|^p \right)^{1/p}$$

In particular:

- ℓ_1 (Manhattan/taxicab) norm:
  $\|v\|_1 = \sum_i |v_i|$
- ℓ_2 (Euclidean) norm:
  $\|v\|_2 = \sqrt{\sum_i v_i^2}$
- ℓ_∞ norm: $\|v\|_\infty = \max_i |v_i|$

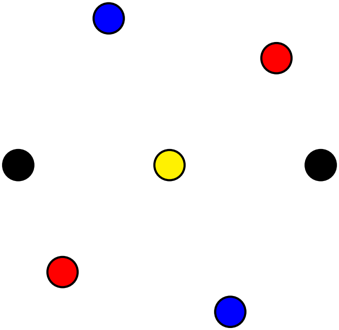


- We call the space $\mathbb{R}^d$, equipped with the ℓ_p norm, just ℓ_p^d.
- Note that these norms can all be computed in time $O(d)$.

The diameter problem

Problem

Given a set S of n points in ℓ_p^d, find a pair $p, q \in S$ such that $\|p - q\|_p$ is maximised.



The diameter problem in ℓ_∞

- Testing every pair of points gives an $O(dn^2)$ algorithm.
- Not great when n is large and d is small (e.g. constant).

The diameter problem in ℓ_∞

- Testing every pair of points gives an $O(dn^2)$ algorithm.
- Not great when n is large and d is small (e.g. constant).
- But in the ∞-norm we can do better!

The diameter problem in ℓ_∞

- Testing every pair of points gives an $O(dn^2)$ algorithm.
- Not great when n is large and d is small (e.g. constant).
- But in the ∞-norm we can do better!

$$\max_{p,q \in S} \|p - q\|_\infty = \max_{p,q \in S} \max_i |p_i - q_i|$$

The diameter problem in ℓ_∞

- Testing every pair of points gives an $O(dn^2)$ algorithm.
- Not great when n is large and d is small (e.g. constant).
- But in the ∞-norm we can do better!

$$\begin{aligned}\max_{p,q\in S} \|p-q\|_\infty &= \max_{p,q\in S} \max_i |p_i - q_i| \\ &= \max_i \left(\max_{p\in S} p_i - \min_{q\in S} q_i \right)\end{aligned}$$

The diameter problem in ℓ_∞

- Testing every pair of points gives an $O(dn^2)$ algorithm.
- Not great when n is large and d is small (e.g. constant).
- But in the ∞-norm we can do better!

$$\begin{aligned}\max_{p,q\in S} \|p-q\|_\infty &= \max_{p,q\in S} \max_i |p_i - q_i| \\ &= \max_i \left(\max_{p\in S} p_i - \min_{q\in S} q_i \right)\end{aligned}$$

- This gives an $O(dn)$ algorithm for computing the diameter in ℓ_∞.
- But what if we want to use (say) the ℓ_1 norm?

From ℓ_1 to ℓ_∞

We'll construct a mapping $f : \mathbb{R}^d \to \mathbb{R}^{d'}$ such that:

- $\|f(p) - f(q)\|_\infty = \|p - q\|_1$
- $d' = 2^d$
- f can be computed in time $O(d2^d)$.

From ℓ_1 to ℓ_∞

We'll construct a mapping $f : \mathbb{R}^d \to \mathbb{R}^{d'}$ such that:

- $\|f(p) - f(q)\|_\infty = \|p - q\|_1$
- $d' = 2^d$
- f can be computed in time $O(d2^d)$.

Implies an $O(nd) + O(nd2^d) = O(nd2^d)$ algorithm for computing the diameter in ℓ_1.

Assuming constant dimension, this is linear time.

From ℓ_1 to ℓ_∞

Our function f is defined elementwise. For each vector $s \in \{-1, 1\}^d$, define

$$f_s(p) = s \cdot p = \sum_{i=1}^{d} s_i p_i$$

Then concatenate all the $f_s(p)$ for the 2^d different s to form $f(p)$.

From ℓ_1 to ℓ_∞

Our function f is defined elementwise. For each vector $s \in \{-1, 1\}^d$, define

$$f_s(p) = s \cdot p = \sum_{i=1}^{d} s_i p_i$$

Then concatenate all the $f_s(p)$ for the 2^d different s to form $f(p)$.

We need to show that $\|f(p) - f(q)\|_\infty = \|p - q\|_1$.

From ℓ_1 to ℓ_∞

Our function f is defined elementwise. For each vector $s \in \{-1, 1\}^d$, define

$$f_s(p) = s \cdot p = \sum_{i=1}^{d} s_i p_i$$

Then concatenate all the $f_s(p)$ for the 2^d different s to form $f(p)$.

We need to show that $\|f(p) - f(q)\|_\infty = \|p - q\|_1$.

- f is linear, so would suffice that $\|f(p - q)\|_\infty = \|p - q\|_1$.

From ℓ_1 to ℓ_∞

Our function f is defined elementwise. For each vector $s \in \{-1, 1\}^d$, define

$$f_s(p) = s \cdot p = \sum_{i=1}^{d} s_i p_i$$

Then concatenate all the $f_s(p)$ for the 2^d different s to form $f(p)$.

We need to show that $\|f(p) - f(q)\|_\infty = \|p - q\|_1$.

- f is linear, so would suffice that $\|f(p - q)\|_\infty = \|p - q\|_1$.
- For any x, $f_s(x)$ is clearly maximised when $s_i = \operatorname{sgn} x_i$ for all i.

From ℓ_1 to ℓ_∞

Our function f is defined elementwise. For each vector $s \in \{-1, 1\}^d$, define

$$f_s(p) = s \cdot p = \sum_{i=1}^{d} s_i p_i$$

Then concatenate all the $f_s(p)$ for the 2^d different s to form $f(p)$.

We need to show that $\|f(p) - f(q)\|_\infty = \|p - q\|_1$.

- f is linear, so would suffice that $\|f(p - q)\|_\infty = \|p - q\|_1$.
- For any x, $f_s(x)$ is clearly maximised when $s_i = \operatorname{sgn} x_i$ for all i.
- But for any x, $\|x\|_1 = \sum_i (\operatorname{sgn} x_i) x_i$.

From ℓ_1 to ℓ_∞

Our function f is defined elementwise. For each vector $s \in \{-1, 1\}^d$, define

$$f_s(p) = s \cdot p = \sum_{i=1}^{d} s_i p_i$$

Then concatenate all the $f_s(p)$ for the 2^d different s to form $f(p)$.

We need to show that $\|f(p) - f(q)\|_\infty = \|p - q\|_1$.

- f is linear, so would suffice that $\|f(p - q)\|_\infty = \|p - q\|_1$.
- For any x, $f_s(x)$ is clearly maximised when $s_i = \operatorname{sgn} x_i$ for all i.
- But for any x, $\|x\|_1 = \sum_i (\operatorname{sgn} x_i) x_i$.
- So for the s such that $s_i = \operatorname{sgn}(p - q)_i$, $f_s(p - q) = \|p - q\|_1$.

Norm embeddings

This map f is an example of an embedding of ℓ_1^d in $\ell_\infty^{2^d}$.

Norm embeddings

This map f is an example of an embedding of ℓ_1^d in $\ell_\infty^{2^d}$.

Definition

Let (X, D) and (Y, D') be metric spaces. A map $f : X \to Y$ is said to be a randomised embedding of X in Y with distortion c and failure probability δ if, for all $p, q \in X$,

$$D(p, q)/c \leq D'(f(p), f(q)) \leq c\, D(p, q)$$

with probability $1 - \delta$.

Norm embeddings

This map f is an example of an embedding of ℓ_1^d in $\ell_\infty^{2^d}$.

Definition

Let (X, D) and (Y, D') be metric spaces. A map $f : X \to Y$ is said to be a randomised embedding of X in Y with distortion c and failure probability δ if, for all $p, q \in X$,

$$D(p,q)/c \leq D'(f(p), f(q)) \leq c\, D(p,q)$$

with probability $1 - \delta$.

Our embedding $\ell_1^d \to \ell_\infty^{2^d}$ is deterministic and has distortion 1 (is isometric)... but we won't always be so lucky.

Dimensionality reduction

- Many problems suffer from the so-called curse of dimensionality: they become exponentially harder as the dimension increases.

Dimensionality reduction

- Many problems suffer from the so-called curse of dimensionality: they become exponentially harder as the dimension increases.
- We can mitigate this by exponentially reducing the dimension using a randomised embedding.

Dimensionality reduction

- Many problems suffer from the so-called curse of dimensionality: they become exponentially harder as the dimension increases.
- We can mitigate this by exponentially reducing the dimension using a randomised embedding.

Johnson-Lindenstrauss Lemma

For any ϵ, and any $d' \leq d$, there is a randomised embedding $\ell_2^d \to \ell_2^{d'}$ with distortion $1 + \epsilon$ and failure probability $e^{\Omega(-d'\epsilon^2)}$.

Dimensionality reduction

- Many problems suffer from the so-called curse of dimensionality: they become exponentially harder as the dimension increases.
- We can mitigate this by exponentially reducing the dimension using a randomised embedding.

Johnson-Lindenstrauss Lemma

For any ϵ, and any $d' \leq d$, there is a randomised embedding $\ell_2^d \to \ell_2^{d'}$ with distortion $1+\epsilon$ and failure probability $e^{\Omega(-d'\epsilon^2)}$.

Corollary

For any ϵ there is a randomised embedding $\ell_2^d \to \ell_2^{O(\log n/\epsilon^2)}$ of n points with distortion $1+\epsilon$ and constant failure probability.

The embedding

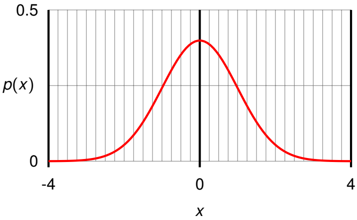
We construct a $d' \times d$ matrix M where each entry of M is random and picked from a normal distribution $N(0, 1)$, rescaled by $1/\sqrt{d'}$.

The embedding

We construct a $d' \times d$ matrix M where each entry of M is random and picked from a normal distribution $N(0, 1)$, rescaled by $1/\sqrt{d'}$.

- This is a continuous probability distribution with probability density function

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}.$$

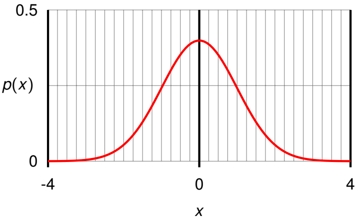


The embedding

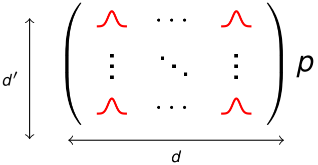
We construct a $d' \times d$ matrix M where each entry of M is random and picked from a normal distribution $N(0,1)$, rescaled by $1/\sqrt{d'}$.

- This is a continuous probability distribution with probability density function

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}.$$



Then set $p' = M p =$

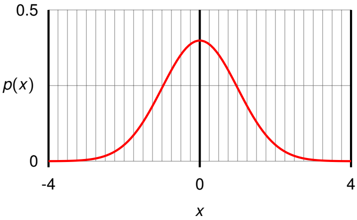


The embedding

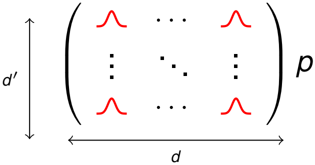
We construct a $d' \times d$ matrix M where each entry of M is random and picked from a normal distribution $N(0, 1)$, rescaled by $1/\sqrt{d'}$.

- This is a continuous probability distribution with probability density function

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}.$$



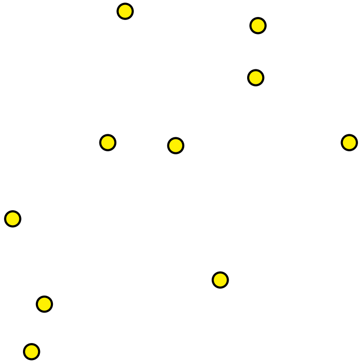
Then set $p' = M\,p =$



This embedding can be performed in $O(d\,d')$ time per point.

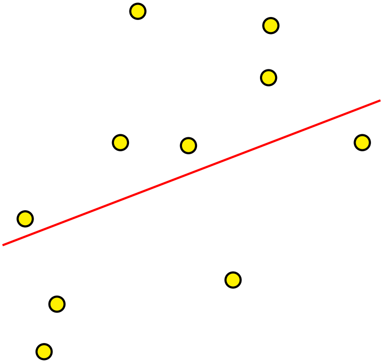
Example

We embed $\ell_2^2 \to \ell_2^1$. This is basically the same as projecting onto a random line.



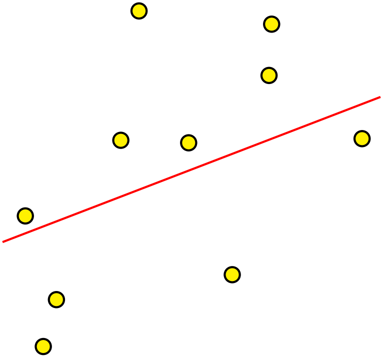
Example

We embed $\ell_2^2 \rightarrow \ell_2^1$. This is basically the same as projecting onto a random line.



Example

We embed $\ell_2^2 \to \ell_2^1$. This is basically the same as projecting onto a random line.



Proof of correctness (sketch)

Johnson-Lindenstrauss Lemma

For any ϵ there is a randomised embedding $\ell_2^d \to \ell_2^{d'}$ with distortion $1 + \epsilon$ and failure probability $e^{\Omega(-d'\epsilon^2)}$.

Proof of correctness (sketch)

Johnson-Lindenstrauss Lemma

For any ϵ there is a randomised embedding $\ell_2^d \to \ell_2^{d'}$ with distortion $1+\epsilon$ and failure probability $e^{\Omega(-d'\epsilon^2)}$.

- We need to show that $\Pr_M[\|Mp - Mq\|_2 > (1+\epsilon)\|p-q\|_2] \leq e^{\Omega(-d'\epsilon^2)}$.

Proof of correctness (sketch)

Johnson-Lindenstrauss Lemma

For any ϵ there is a randomised embedding $\ell_2^d \to \ell_2^{d'}$ with distortion $1+\epsilon$ and failure probability $e^{\Omega(-d'\epsilon^2)}$.

- We need to show that $\Pr_M[\|Mp - Mq\|_2 > (1+\epsilon)\|p-q\|_2] \leq e^{\Omega(-d'\epsilon^2)}$.
- Suffices to show that, for all x, $\Pr_M[\|Mx\|_2 > (1+\epsilon)\|x\|_2] \leq e^{\Omega(-d'\epsilon^2)}$: i.e. that the expected resulting length of a vector is sharply concentrated about its mean.

Proof of correctness (sketch)

Johnson-Lindenstrauss Lemma

For any ϵ there is a randomised embedding $\ell_2^d \to \ell_2^{d'}$ with distortion $1+\epsilon$ and failure probability $e^{\Omega(-d'\epsilon^2)}$.

- We need to show that $\Pr_M[\|Mp - Mq\|_2 > (1+\epsilon)\|p-q\|_2] \leq e^{\Omega(-d'\epsilon^2)}$.
- Suffices to show that, for all x, $\Pr_M[\|Mx\|_2 > (1+\epsilon)\|x\|_2] \leq e^{\Omega(-d'\epsilon^2)}$: i.e. that the expected resulting length of a vector is sharply concentrated about its mean.
- Set $L = \|v\|_2^2$, $L' = \|Mv\|_2^2$. Then $\mathbb{E}[L'] = L$.

Proof of correctness (sketch)

Johnson-Lindenstrauss Lemma

For any ϵ there is a randomised embedding $\ell_2^d \to \ell_2^{d'}$ with distortion $1+\epsilon$ and failure probability $e^{\Omega(-d'\epsilon^2)}$.

- We need to show that $\Pr_M[\|Mp - Mq\|_2 > (1+\epsilon)\|p-q\|_2] \leq e^{\Omega(-d'\epsilon^2)}$.
- Suffices to show that, for all x, $\Pr_M[\|Mx\|_2 > (1+\epsilon)\|x\|_2] \leq e^{\Omega(-d'\epsilon^2)}$: i.e. that the expected resulting length of a vector is sharply concentrated about its mean.
- Set $L = \|v\|_2^2$, $L' = \|Mv\|_2^2$. Then $\mathbb{E}[L'] = L$. Also, for any $\beta > 1$,
  - $\Pr[L' > \beta L] < O(L')\, e^{-\Omega(L'\beta^2)}$, and
  - $\Pr[L' < L/\beta] < O(L')\, e^{-\Omega(L'/\beta^2)}$

Notes on the J-L Lemma

- This result was proven in 1984 and has been rediscovered several times, e.g. in the applied setting by Kaski in 1998.

Notes on the J-L Lemma

- This result was proven in 1984 and has been rediscovered several times, e.g. in the applied setting by Kaski in 1998.
- Ailon and Chazelle have recently given a version of this embedding that uses $\approx O(d \log d)$ time per vector (rather than $O(d \log^{O(1)} n)$)

Notes on the J-L Lemma

- This result was proven in 1984 and has been rediscovered several times, e.g. in the applied setting by Kaski in 1998.
- Ailon and Chazelle have recently given a version of this embedding that uses $\approx O(d \log d)$ time per vector (rather than $O(d \log^{O(1)} n)$)
- The fact that the elements of the matrix M are normally distributed isn't important: in fact you can put almost anything in M – e.g. random ± 1 entries (easier to implement).

Notes on the J-L Lemma

- This result was proven in 1984 and has been rediscovered several times, e.g. in the applied setting by Kaski in 1998.
- Ailon and Chazelle have recently given a version of this embedding that uses $\approx O(d \log d)$ time per vector (rather than $O(d \log^{O(1)} n)$)
- The fact that the elements of the matrix M are normally distributed isn't important: in fact you can put almost anything in M – e.g. random ± 1 entries (easier to implement).
- This is an example of the concentration of measure phenomenon (random variables in high dimensions are concentrated around their means).

Applications

- Imagine we have a problem to solve with n points in d dimensions, and an algorithm with running time $T(n, d)$.

Applications

- Imagine we have a problem to solve with n points in d dimensions, and an algorithm with running time $T(n, d)$.
- Thanks to the J-L Lemma, this gives an approximation algorithm that runs in time $T(n, \log^{O(1)} n) + O(nd \log^{O(1)} n)$

Applications

- Imagine we have a problem to solve with n points in d dimensions, and an algorithm with running time $T(n, d)$.
- Thanks to the J-L Lemma, this gives an approximation algorithm that runs in time $T(n, \log^{O(1)} n) + O(nd \log^{O(1)} n)$
- This might be a polynomial or exponential speed-up, depending on T.

Example:

Applications

- Imagine we have a problem to solve with n points in d dimensions, and an algorithm with running time $T(n, d)$.
- Thanks to the J-L Lemma, this gives an approximation algorithm that runs in time $T(n, \log^{O(1)} n) + O(nd \log^{O(1)} n)$
- This might be a polynomial or exponential speed-up, depending on T.

Example:

- Closest pair/diameter: $O(n^2 d)$ time $\rightarrow$ $O((n^2 \log^{O(1)} n + nd \log n)/\epsilon^2)$ time.

Applications

- Imagine we have a problem to solve with n points in d dimensions, and an algorithm with running time $T(n, d)$.
- Thanks to the J-L Lemma, this gives an approximation algorithm that runs in time $T(n, \log^{O(1)} n) + O(nd \log^{O(1)} n)$
- This might be a polynomial or exponential speed-up, depending on T.

Example:

- Closest pair/diameter: $O(n^2 d)$ time $\rightarrow$ $O((n^2 \log^{O(1)} n + nd \log n)/\epsilon^2)$ time.

This result can also be used for clustering high-dimensional data: performance is similar to Principal Components Analysis (PCA) and it's easier to implement.

Nearest neighbour search

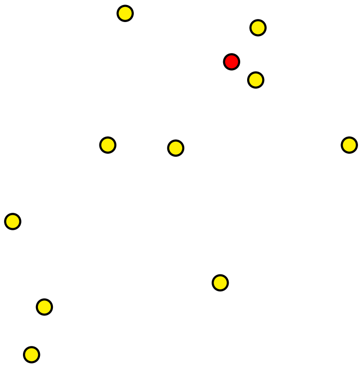
Problem: Pre-process a set of points S in $\mathbb{R}^d$ so that queries of the following sort can be answered efficiently:

- Given a point x, what is the nearest neighbour of x in S?

Nearest neighbour search

Problem: Pre-process a set of points S in $\mathbb{R}^d$ so that queries of the following sort can be answered efficiently:

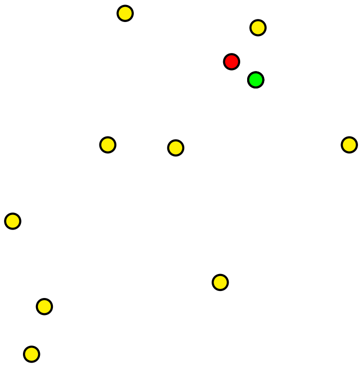
- Given a point x, what is the nearest neighbour of x in S?



Nearest neighbour search

Problem: Pre-process a set of points S in $\mathbb{R}^d$ so that queries of the following sort can be answered efficiently:

- Given a point x, what is the nearest neighbour of x in S?



Approximate nearest neighbour search

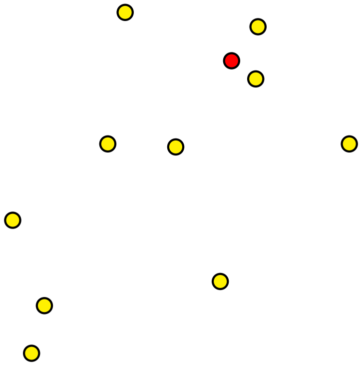
Problem: Pre-process a set of points S in $\mathbb{R}^d$ so that queries of the following sort can be answered efficiently:

- Given a point x, whose nearest neighbour in S is y, output any point z in S such that $\|x - z\| \leq c\|x - y\|$.

Approximate nearest neighbour search

Problem: Pre-process a set of points S in $\mathbb{R}^d$ so that queries of the following sort can be answered efficiently:

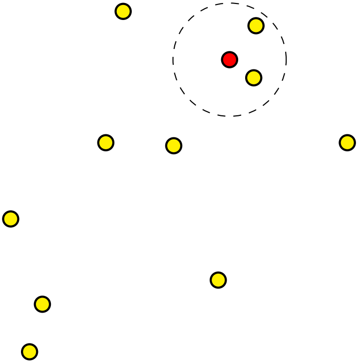
- Given a point x, whose nearest neighbour in S is y, output any point z in S such that $\|x - z\| \leq c\|x - y\|$.



Approximate nearest neighbour search

Problem: Pre-process a set of points S in $\mathbb{R}^d$ so that queries of the following sort can be answered efficiently:

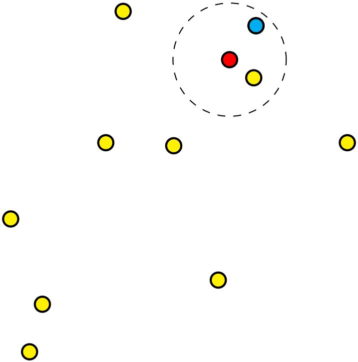
- Given a point x, whose nearest neighbour in S is y, output any point z in S such that $\|x - z\| \leq c\|x - y\|$.



Approximate nearest neighbour search

Problem: Pre-process a set of points S in $\mathbb{R}^d$ so that queries of the following sort can be answered efficiently:

- Given a point x, whose nearest neighbour in S is y, output any point z in S such that $\|x - z\| \leq c\|x - y\|$.



Approximate nearest neighbour search

- The exact nearest-neighbour problem appears to be intractable (conjecture: any data structure achieving poly(d) query time must use space super-polynomial in n).

Approximate nearest neighbour search

- The exact nearest-neighbour problem appears to be intractable (conjecture: any data structure achieving poly(d) query time must use space super-polynomial in n).
- Indyk and Motwani recently (1998) proposed a more efficient approximate nearest neighbour search algorithm.

Approximate nearest neighbour search

- The exact nearest-neighbour problem appears to be intractable (conjecture: any data structure achieving poly(d) query time must use space super-polynomial in n).
- Indyk and Motwani recently (1998) proposed a more efficient approximate nearest neighbour search algorithm.
- Based on an algorithm which finds a $(1+\epsilon)$ nearest neighbour in poly$(d, \log n, 1/\epsilon)$ time using a data structure of size $O(1/\epsilon)^d n \,\mathrm{polylog}(n)$.

Approximate nearest neighbour search

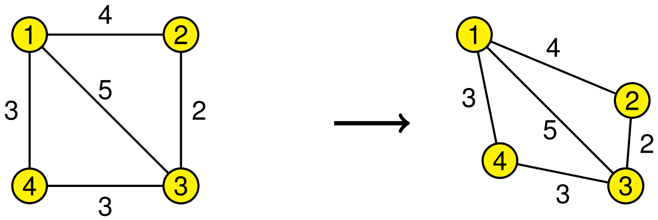
- The exact nearest-neighbour problem appears to be intractable (conjecture: any data structure achieving poly(d) query time must use space super-polynomial in n).
- Indyk and Motwani recently (1998) proposed a more efficient approximate nearest neighbour search algorithm.
- Based on an algorithm which finds a $(1+\epsilon)$ nearest neighbour in poly($d, \log n, 1/\epsilon$) time using a data structure of size $O(1/\epsilon)^d n \,\mathrm{polylog}(n)$.
- The J-L Lemma allows this space bound to be reduced to $n^{O(\log(1/\epsilon)/\epsilon^2)}$ – an exponential reduction.

Finite metric embeddings

- We now consider embeddings of finite metrics in norms.
- This can be visualised as mapping a graph into a vector space.

Finite metric embeddings

- We now consider embeddings of finite metrics in norms.
- This can be visualised as mapping a graph into a vector space.
- For example, consider the following isometric embedding of a graph into ℓ_2^2:



Any finite metric can be embedded in ℓ_∞

Theorem (Frèchet)

Any finite metric space (X, D) with $|X| = n$ embeds isometrically into ℓ_∞^n.

Any finite metric can be embedded in ℓ_∞

Theorem (Frèchet)

Any finite metric space (X, D) with $|X| = n$ embeds isometrically into ℓ_∞^n.

Proof:

- Enumerate X as $X = \{x_1, \ldots, x_n\}$. Then our embedding will be the mapping $f : X \to \mathbb{R}^n$ where $f(p)_i = D(p, x_i)$.

Any finite metric can be embedded in ℓ_∞

Theorem (Frèchet)

Any finite metric space (X, D) with $|X| = n$ embeds isometrically into ℓ_∞^n.

Proof:

- Enumerate X as $X = \{x_1, \dots, x_n\}$. Then our embedding will be the mapping $f : X \to \mathbb{R}^n$ where $f(p)_i = D(p, x_i)$.
- For any $p, q \in X$, $\|f(p) - f(q)\|_\infty = \max_i |f(p)_i - f(q)_i| = \max_i |D(p, x_i) - D(q, x_i)| \leq D(p, q)$ ("reverse" triangle inequality)

Any finite metric can be embedded in ℓ_∞

Theorem (Frèchet)

Any finite metric space (X, D) with $|X| = n$ embeds isometrically into ℓ_∞^n.

Proof:

- Enumerate X as $X = \{x_1, \dots, x_n\}$. Then our embedding will be the mapping $f : X \to \mathbb{R}^n$ where $f(p)_i = D(p, x_i)$.
- For any $p, q \in X$, $\|f(p) - f(q)\|_\infty = \max_i |f(p)_i - f(q)_i| = \max_i |D(p, x_i) - D(q, x_i)| \leq D(p, q)$ ("reverse" triangle inequality)
- On the other hand, $|D(p, p) - D(q, p)| = D(q, p)$, so $\|f(p) - f(q)\|_\infty = D(p, q)$.

Embedding into ℓ_2

- This embedding uses *n* dimensions.
- If we allow some distortion, we can do exponentially better.

Embedding into ℓ_2

- This embedding uses n dimensions.
- If we allow some distortion, we can do exponentially better.

Theorem (Bourgain)

For any $1 \leq p \leq 2$, any finite metric space (X, D) with $|X| = n$ can be embedded into $\ell_p^{O(\log^2 n)}$ with distortion $O(\log n)$ in time $O(n^2 \log n)$.

Embedding into ℓ_2

- This embedding uses n dimensions.
- If we allow some distortion, we can do exponentially better.

Theorem (Bourgain)

For any $1 \leq p \leq 2$, any finite metric space (X, D) with $|X| = n$ can be embedded into $\ell_p^{O(\log^2 n)}$ with distortion $O(\log n)$ in time $O(n^2 \log n)$.

The proof is based on similar ideas, but is more complex and involves replacing the points x_i by sets of points.

Applications

An easy application: efficient storage & computation of shortest paths.

Applications

An easy application: efficient storage & computation of shortest paths.

- Say a graph G can be embedded in ℓ_p^d with distortion c for some p, d.

Applications

An easy application: efficient storage & computation of shortest paths.

- Say a graph G can be embedded in ℓ_p^d with distortion c for some p, d.
- Then we can store the graph in space $O(nd)$ and c-approximate the shortest path between any two vertices in $O(d)$ time.

Applications

An easy application: efficient storage & computation of shortest paths.

- Say a graph G can be embedded in ℓ_p^d with distortion c for some p, d.
- Then we can store the graph in space $O(nd)$ and c-approximate the shortest path between any two vertices in $O(d)$ time.
- e.g. imagine $d = O(\log n)$: we get space $O(n \log n)$, query time $O(\log n)$.

Sparsest cut

A more complex example: approximating the sparsest cut of a graph.

Sparsest cut

A more complex example: approximating the sparsest cut of a graph.

Problem. Given a graph $G = (V, E)$ with $|V| = n$, find the minimum, over all cuts $V = S \cup T$, of

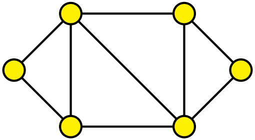
$$\phi(S) = \frac{|E(S, T)|}{|S||T|}$$

Sparsest cut

A more complex example: approximating the sparsest cut of a graph.

Problem. Given a graph $G = (V, E)$ with $|V| = n$, find the minimum, over all cuts $V = S \cup T$, of

$$\phi(S) = \frac{|E(S, T)|}{|S||T|}$$

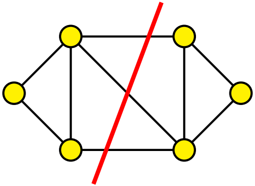


Sparsest cut

A more complex example: approximating the sparsest cut of a graph.

Problem. Given a graph $G = (V, E)$ with $|V| = n$, find the minimum, over all cuts $V = S \cup T$, of

$$\phi(S) = \frac{|E(S, T)|}{|S||T|}$$

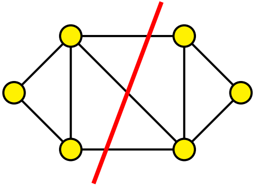


Sparsest cut

A more complex example: approximating the sparsest cut of a graph.

Problem. Given a graph $G = (V, E)$ with $|V| = n$, find the minimum, over all cuts $V = S \cup T$, of

$$\phi(S) = \frac{|E(S, T)|}{|S||T|}$$



"A minimum cut that favours balanced partitions". NP-hard to compute.

Approximating the sparsest cut

Theorem (Linial, London and Rabinovich)

The sparsest cut can be approximated to within an $O(\log n)$ factor in polynomial time.

Approximating the sparsest cut

Theorem (Linial, London and Rabinovich)

The sparsest cut can be approximated to within an $O(\log n)$ factor in polynomial time.

Proof idea:

- Every cut defines a (semi-)metric where $D(x, y) = 1$ if x and y are separated by the cut, and 0 otherwise.

Approximating the sparsest cut

Theorem (Linial, London and Rabinovich)

The sparsest cut can be approximated to within an $O(\log n)$ factor in polynomial time.

Proof idea:

- Every cut defines a (semi-)metric where $D(x, y) = 1$ if x and y are separated by the cut, and 0 otherwise.
- Embed the (unknown!) optimal cut metric in ℓ_1, losing at most $O(\log n)$ in the process.

Approximating the sparsest cut

Theorem (Linial, London and Rabinovich)

The sparsest cut can be approximated to within an $O(\log n)$ factor in polynomial time.

Proof idea:

- Every cut defines a (semi-)metric where $D(x, y) = 1$ if x and y are separated by the cut, and 0 otherwise.
- Embed the (unknown!) optimal cut metric in ℓ_1, losing at most $O(\log n)$ in the process.
- The resulting optimisation problem can be solved efficiently by linear programming.

Exercises

There are many interesting problems related to embedding graphs in norms. Here are some things to think about over the holidays (in rough order of difficulty)...

Exercises

There are many interesting problems related to embedding graphs in norms. Here are some things to think about over the holidays (in rough order of difficulty)...

Prove that:

1. A tree with n vertices can be isometrically embedded into ℓ_1^{n-1}.

Exercises

There are many interesting problems related to embedding graphs in norms. Here are some things to think about over the holidays (in rough order of difficulty)...

Prove that:

1. A tree with n vertices can be isometrically embedded into ℓ_1^{n-1}.
2. The complete graph on n vertices can be isometrically embedded into $\ell_\infty^{\lceil \log_2 n \rceil}$, and this is optimal.

Exercises

There are many interesting problems related to embedding graphs in norms. Here are some things to think about over the holidays (in rough order of difficulty)...

Prove that:

1. A tree with n vertices can be isometrically embedded into ℓ_1^{n-1}.
2. The complete graph on n vertices can be isometrically embedded into $\ell_\infty^{\lceil \log_2 n \rceil}$, and this is optimal.
3. A cycle on n vertices cannot be embedded in a tree with distortion lower than $n-1$.

Exercises

There are many interesting problems related to embedding graphs in norms. Here are some things to think about over the holidays (in rough order of difficulty)...

Prove that:

1. A tree with n vertices can be isometrically embedded into ℓ_1^{n-1}.
2. The complete graph on n vertices can be isometrically embedded into $\ell_\infty^{\lceil \log_2 n \rceil}$, and this is optimal.
3. A cycle on n vertices cannot be embedded in a tree with distortion lower than $n-1$.
4. A complete binary tree on n vertices can be embedded into ℓ_2^n with distortion $O(\sqrt{\log \log n})$.

Conclusion

- Metric embeddings are a powerful tool for finding efficient algorithms.

Conclusion

- Metric embeddings are a powerful tool for finding efficient algorithms.
- Some high-dimensional problems in the ℓ_2 norm can be solved exponentially more quickly using the Johnson-Lindenstrauss Lemma.

Conclusion

- Metric embeddings are a powerful tool for finding efficient algorithms.
- Some high-dimensional problems in the ℓ_2 norm can be solved exponentially more quickly using the Johnson-Lindenstrauss Lemma.
- Any metric on n points can be embedded into $\ell_2^{O(\log^2 n)}$ with $O(\log n)$ distortion.

Conclusion

- Metric embeddings are a powerful tool for finding efficient algorithms.
- Some high-dimensional problems in the ℓ_2 norm can be solved exponentially more quickly using the Johnson-Lindenstrauss Lemma.
- Any metric on n points can be embedded into $\ell_2^{O(\log^2 n)}$ with $O(\log n)$ distortion.

There are many interesting open problems in the field of metric embeddings:

- Mathematical questions
- Theoretical CS
- Applications
- Implementation

Further reading

- "Algorithmic applications of geometric embeddings" by Piotr Indyk.
- "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions" by Alexandr Andoni and Piotr Indyk.
- Several lecture courses: search for "metric embeddings".
- "The geometry of graphs and some of its algorithmic applications" by Linial, London and Rabinovich.

Thanks and Merry Christmas!