proximity problems & curse of dimensionality

- Canonical Proximity Problems
  - Distance Matrix Computation
  - k-nearest Neighbor Problem
  - Fixed-radius nearest neighbors
- Applications
  - Duplicate Detection
  - Image Completion
  - k-NN classification/Regression
  - Collaborative Filtering
  - Search Engine Autocorrect
- Voronoi Diagram and kd Tree
- Processing and Storage
- Data Sparsity
- Issues for Nearest Neighbors
  - Huge Search Space
  - Diminishing volume of n-ball
  - Nearest neighbor instability
- Distance Concentration
- Angle Concentration
  - Generating Random Direction

Imdad ullah Khan

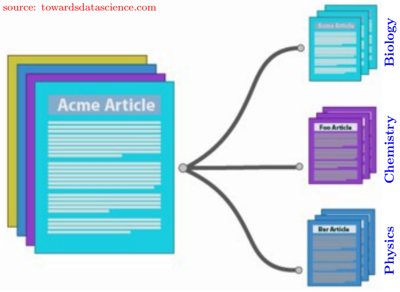
Proximity Problems on High Dimensional Data

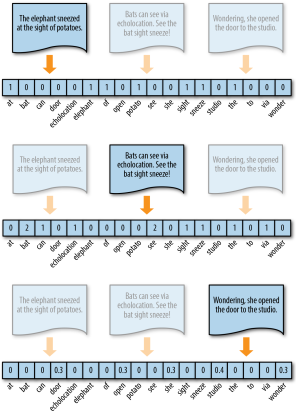
High Dimensional Data

- Text represented as set or bag or TF-IDF of words
- 1000's of unigram, millions of bigrams plus contextual attributes

Bengfort,, Bilbro & Ojeda: Applied Text Analysis with Python

source: towardsdatascience.com





High Dimensional Data

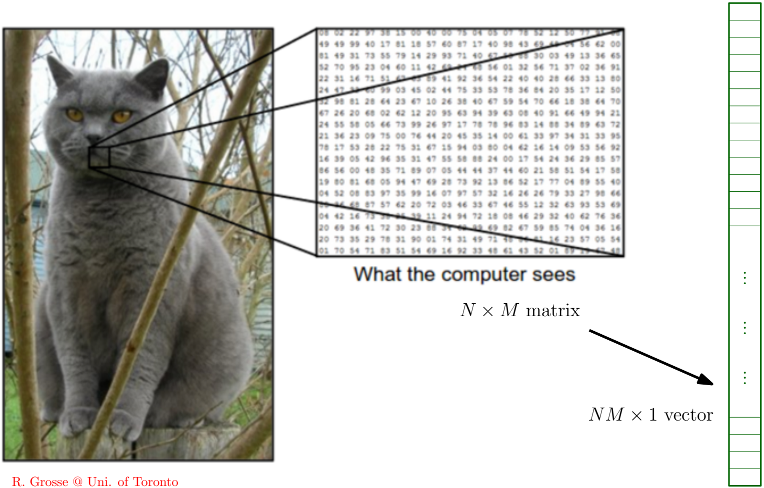
- Utility matrix for recommenders (Amazon product catalogue)
- The netflix prize training set: $\sim 1M$ ratings of the form $\langle$user, movie, date of grade, grade$\rangle$

480,189 users, 17,770 movies

	p_1	p_2	p_3						p_j									p_m
u_1	1		2	1		4			2		3	2		5				2
u_2		1				2		1			2			1				3
u_3		1	1	2				1							1		2	
				3		2			5			2			3	4		
		1			2									5				
u_i			3	2	1		4	5	$\boxed{?}$	1		3	1		2			1
			4													4		
			5			1										5		
		1		4							1	3		5		1	2	
u_n			3		1	1		2	1					4				5

High Dimensional Data

- Images and videos from multi-mega pixels digital cameras



R. Grosse @ Uni. of Toronto

High Dimensional Data

- Social networks as adjacency matrix
- A row of Facebook graph's adjacency matrix has more than a billion dimensions

Proximity Problems

Given a set X of m-dim vectors, with $|X| = n$

Two generic proximity computation problems are building blocks of almost all data analytics

1. Distance Matrix Computation
   - Find $n \times n$ matrix with all pairwise distances
2. k-nearest neighbors (k-NN) problem
   - Given a query point q in the same space as X, return the k closest points in X to q

Proximity Problems: Fixed Radius Nearest Neighbros

Given a set X of m-dim vectors, with $|X| = n$

k-nearest neighbors (k-NN) problem

- Given a query point q in the same space as X, return the k closest points in X to q

A variant of the k-NN problem is

Fixed radius nearest neighbors problem

- Given a query point q in the same space as X and a radius $r > 0$, find all points in X to within radius r from q

This variant is the same as the k-NN problem, in the sense that they are reducibile to each other

Proximity Problems: Applications

Applications: Near Duplicate Detection

Given a set X of m-dim vectors, with $|X| = n$

- Distance Matrix: $n \times n$ matrix with all pairwise distances

Near-duplicates detection

- Find all pairs of points with distance less than δ, or all pairs with distance less than 2σ from the mean distance

News Aggregation, Mirror webpages, Plagiarism Detection

- A story written by one journalist appears differently on many websites
  - different spacing, added advertisements and differences in metadata
- Find such articles for news aggregation site e.g. Google news

Applications: Agglomerative Clustering

Given a set X of m-dim vectors, with $|X| = n$

- Distance Matrix: $n \times n$ matrix with all pairwise distances

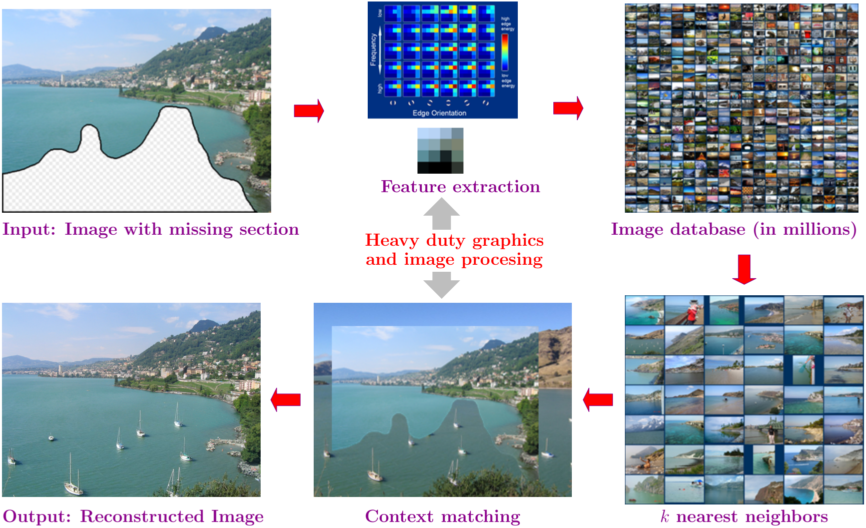
The distance matrix is input for

- Agglomerative clustering
- Principal Component Analysis
- Spectral Clustering
- Multi-dimensional Scaling

Applications: Image Completion

Image Completion, Scene completion, image or art restoration

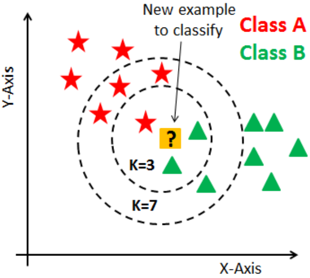
Hays and Efros , Scene Completion Using Millions of Photographs, SIGGRAPH 2007



Applications: kNN Classification

k-NN is a simple method used for classification

The class label of a test instance x is predicted to be the most common class among the k nearest neighbors of x in the train set

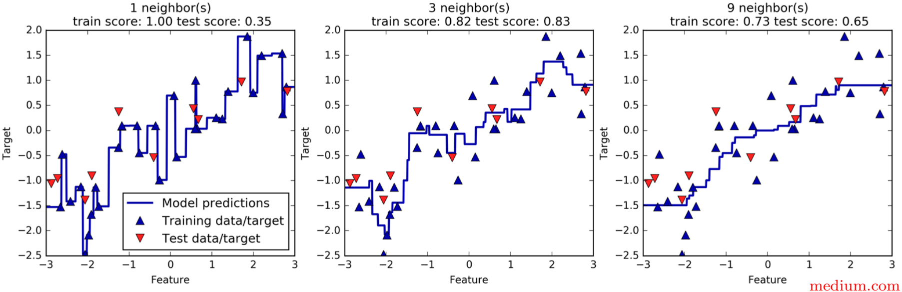


k-NN Classifier

- Assign the test instance (?) class A (★) or class B (▲)
- $k = 3$ nearest neighbors (ℓ_2 distance)
  1 ★ and 2 ▲ $\implies$ assigned label = ▲
- $k = 7$ nearest neighbors (ℓ_2 distance)
  4 ★ and 3 ▲ $\implies$ assigned label = ★

Applications: kNN Regression

In k-NN regression value of the target variable y for an instance $\mathbf{x}$ is estimated as average of y's values of the k instance that are nearest to $\mathbf{x}$

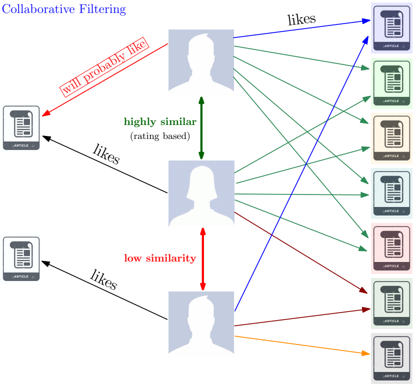


medium.com

Applications: Collaborative Filtering

Collaboratively filter (personalize) ratings using only the rating matrix U

- Find the set N of users with similar ratings as of i
  - Find the top k similar rows to the ith row
- Estimate $U(i,j)$ as an "average" of $U(a,j)$'s for $a \in N$

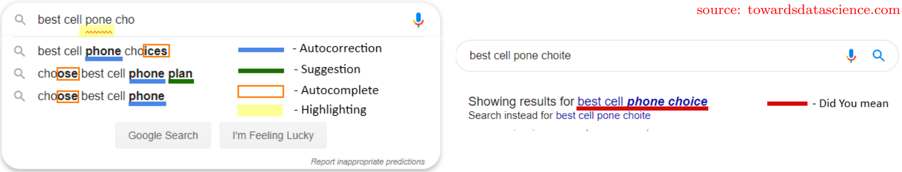


Applications: Autocorrect utility

Search Engines' Autocorrect utility

- On a query phrase q, find the most similar query phrases in a dataset
- Has to be done in near real-time

source: towardsdatascience.com



Lateral Phishing Emails:

- Phishing emails sent from a legitimate but compromised email address
- Checking if recipient list is very dissimilar from usual recipients

Approaches for kNN problem

Brute Force Algorithms

Given a set X of m-dim vectors, with $|X| = n$

Almost all $d(x, y)$ measures require traversal of all coordinates of x and y

Runtime of the brute force algorithm for D matrix computation

$$O(n^2 \times m)$$

Runtime of the brute force algorithm for k-$NN(q)$ is

$$O(n \times m)$$

Runtimes grows linearly with dimensionality and quadratically or linearly with number of points

In dimensionality reduction we dea with the factor of m, here we deal with the factor n

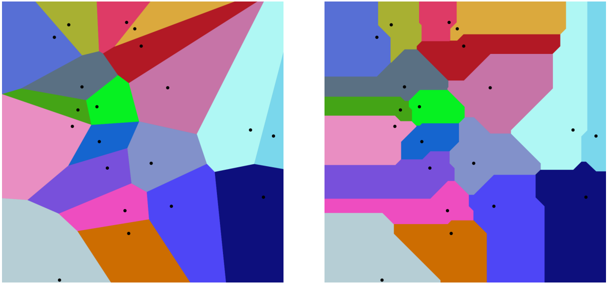
List

- Store X in a list
- No preprocessing
- On query run a FINDMIN algorithm on distance to q
- Runtime is $O(n)$ distance computations

- For $m = 1$, store X in a sorted array
- Best data structure for 1-d $NN(q)$
- With Binary search for q runtime is $O(\log n)$ distance computations

Voronoi Diagrams

- Voronoi diagram ($m = 2$) Partition of plane into nearest neighbor regions
- Region R_i of a point $x_i \in X$ is the set of all points that are NN of x_i
- R_i: intersection of perp. bisectors of segments b/w x_i and other points
- For $m = 2$, Fortune's algorithm for voronoi diagram in $O(n \log n)$

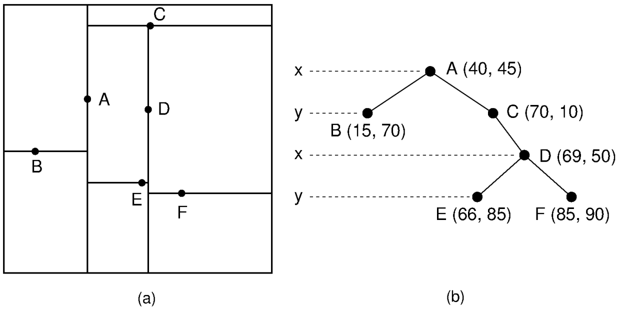


Voronoi diagrams of 20 points under (left) Euclidean and (right) Manhattan distance. source: Wikipedia

Hard to even describe in higher dimensions

kd-Tree

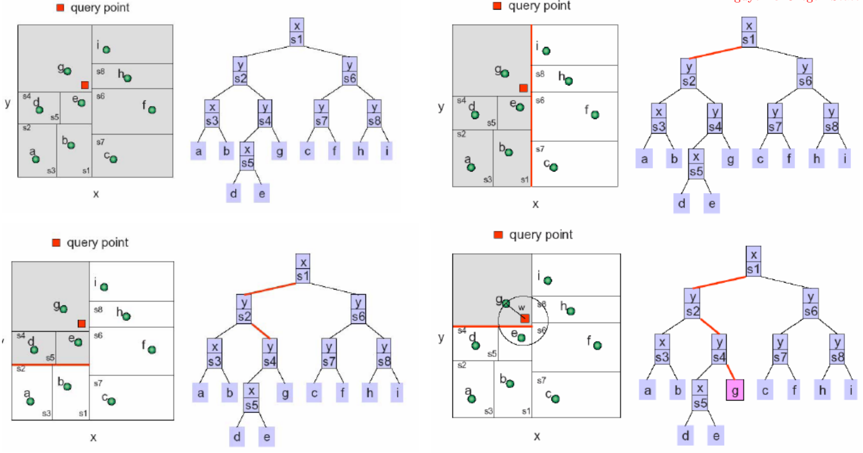
- *kd*-tree data structure: Partition the space into non-uniform cells
- A binary tree where each level compare 1 dimension (cutting dimension)
- Internal nodes correspond to hyperplanes splitting space in 2 half spaces
- Halve the points by a hyperplane perpendicular to one dimension
- Recursively construct *kd*-tree for the two halves, until one point remains
- Cycle through all dimensions



kd-Tree

Searching for nearest neighbor in *kd*-tree

T. Nguyen @ Oregon State

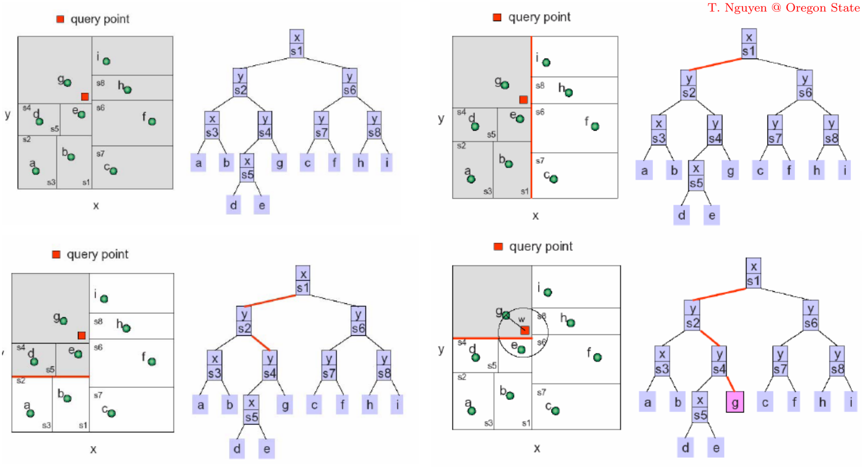


kd-trees are very effective for dimensions ≤ 10 or so

kd-Tree

Searching for nearest neighbor in *kd*-tree

T. Nguyen @ Oregon State



kd-trees are very effective for dimensions ≤ 10 or so

Curse of dimensionality

Curse of dimensionality

Richard Bellman coined the phrase, referring to difficulty of dynamic optimization with many variables

Broadly, we face these issues when working with high dimensional data

- Computational challenging, processing, storing, communication
- In general as number of features increases redundancy also increases
  - More noise added to data than signal
  - Quality of Analytics degrades
- Hard to visualize and interpret

Aspects of Curse of dimensionality

Issues with Higher Dimensional Data

- Computational and Storage Challenges
  - Complexity of exact algorithms for proximity computation problems
- Data Sparsity (Sparse training set generalization)
- Issues for Nearest Neighbors
  - Huge Search Space
  - Diminishing volume of n-ball
  - Stability of nearest neighbors
- Distance Concentration
- Angle Concentration

Computational Complexity

Given a set X of m-dim vectors, with $|X| = n$

Almost all $d(x, y)$ measures require traversal of all coordinates of x and y

Runtime of the brute force algorithms for D matrix computation

$$O(n^2 \times m)$$

Runtime of the brute force algorithms for k-$NN(q)$ is

$$O(n \times m)$$

Both runtimes grow linearly with dimensionality

Data Sparsity

As dimensionality increases the relative input space covered by a fixed-size training set diminishes

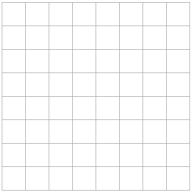
Many methods require a sizeable number of examples/samples in every region of the space to support a hypothesis or train a generalizable model

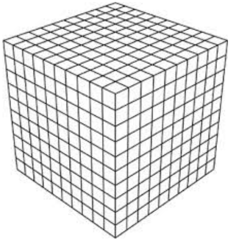
1000 students (discretized) scores in course $\in \{0, 25, 50, 75, 100\}\%$

- Two courses c_1 and $c_2 \rightarrow 5 \times 5$ grade combinations
  - Each combination has average $^{1000}/_{25} = 40$ students
  - Good enough sample size, can infer rules like
  - if $grade(c_1) \leq 50 \wedge grade(c_2) \geq 75$, then student is Math major
- For 4 course, number of grade combinations is $5^4 = 625$
  - 1.6 students per combination
- For 10 course, average students per combination is 0.0001024
  - Almost all combinations are never observed

Huge Search Space for Nearest neighbor

- For large dimensions partition the space into cells (grids or mesh)
- Search for kNN in the cell containing query q and 'neighboring' cells
- Number of 'neighboring' cells in 2-d is $3^2 = 9$, in 3-d 3^3, in m-d, 3^m





Grid can be non-uniform as in kd-tree

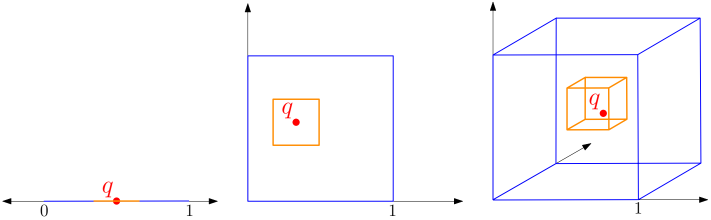
Huge Search Space for Nearest neighbor

Another way to look at this

Higher dimensional neighborhood is very large and not local

▷ The notion of nearest neighbor breaks down

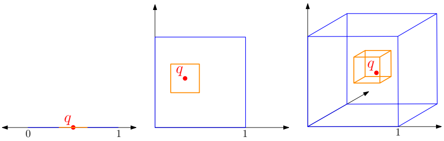
- Suppose n points are placed uniformly at random in $[0,1]^m$
- Grow a hypercube around q to contain f fraction of points ($k = fn$)
- Expected edge length ℓ : $E_m(\ell) = f^{1/m}$
- In $10d$ to get 10% points around q need a cube with edge length 0.8
- To get only 1% point need to extend cube 0.63 along each dimension



Huge Search Space for Nearest neighbor

Another way to look at non-locality of higher dimensional neighborhoods

- Suppose 5000 points are randomly placed in $[0,1]^m$. Let $q = \mathbf{0}$
- In 1d must go a distance $5/5000 = 0.001$ on average to capture 5 NN
- In 2d must go $\sqrt{5/5000} = 0.031$ units along both dimensions
- In 3d must go $\sqrt[3]{0.001} = 0.1 = 10\%$ of the total (unit) length
- In 4d must go $\sqrt[4]{0.001} = 0.177 = 17.7\%$ of unit length
- In 10d must go 50.1% of unit length along each dimension
- In md must go $(5/5000)^{1/m}$ along each dimension



In high dimensional space nobody can hear you scream

Diminishing Volume of m-ball

A manifestation of this phenomenon that points in higher dimensions are isolated is the diminishing relative volume of the m-ball in m-cube

The m-ball (m-dim hypersphere) of radius r centered at origin

$$B_{m,r} := \left\{\mathbf{x} \in \mathbb{R}^m : d(\mathbf{x}, \mathbf{0} \leq r) \implies \|\mathbf{x}\|_2 \leq r\right\}$$

$$\text{Volume of } B_{m,r} : \qquad V_m(r) = \frac{\pi^{m/2}}{\Gamma(m/2 + 1)}\, r^m$$

$\Gamma(\cdot)$ essentially is factorial of fractional numbers

$$V_m(r) = \frac{\pi^{m/2}}{m/2!} r^m \qquad \text{For simplicity assume } m \text{ is even}$$

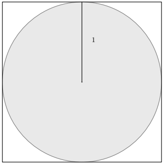
The m-cube (m-d hypercube) is the set $[-1, 1]^m$ (note edge length is 2)

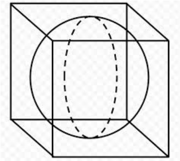
$$\text{Volume of } m\text{-cube:} \qquad 2^m$$

Diminishing Volume of m-ball

In m-d ratio of volume of unit m-Ball to that of m-cube (edge length 2)

$$\frac{\pi^{m/2}/_{m/2!}}{2^m}$$ approaches 0 very fast



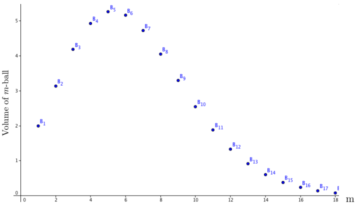


dim m	volume of m-ball	volume of m-cube	ratio
2	π	2^2	~ 0.785
3	$4\pi/3$	2^3	~ 0.523
4	$\pi^2/2$	2^4	~ 0.308
6	$\pi^3/6$	2^6	~ 0.080
m	$\frac{\pi^{m/2}}{m/2!}$	2^m	$\rightarrow 0$

Diminishing Volume of m-ball

Ratio of volumes of unit m-Ball and $[-1,1]^m$

$$\frac{\pi^{m/2}/_{m/2!}}{2^m}$$

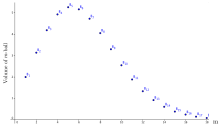


- In higher dimensions all the volume is in 'corners'
- Points in high dimensional spaces are isolated (empty surrounding)
- The probability that a randomly generated point is within r radius of q approaches 0 as dimensionality increases
- The probability of a close nearest neighbor in a data set is very small
- Caveat: Real datasets are not random
- Overcome this by getting larger training set (exponential in m)

Diminishing Volume of m-ball

ratio of volumes of unit m-Ball and $[-1,1]^m$

$$\frac{\pi^{m/2}/m/2!}{2^m}$$



- In higher dimensions all the volume is in 'corners'
- Probability of a close nearest neighbor in random data set is very small
- Overcome this by getting larger training set (exponential in m)

To cover $[-1,1]^m$ with $B_{m,1}$'s, the number of balls n must be

$$n \geq \frac{2^m}{V_m(1)} = \frac{2^m}{\pi^{m/2}/m/2!} = \frac{m/2!\, 2^m}{\pi^{m/2}} \overset{m\to\infty}{\sim} \sqrt{m\pi}\left(\frac{m2^{m/2}}{2\pi e}\right)^{m/2}$$

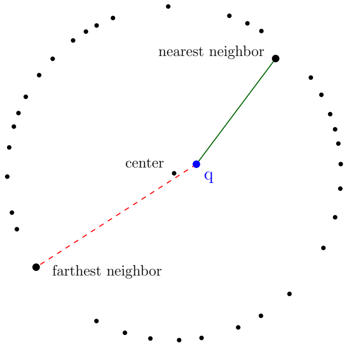
For $m = 16$ (a very small number) this n is substantially bigger than 2^{58}

Instability of Nearest neighbor

In higher dimension the notion of nearest neighbor breaks down

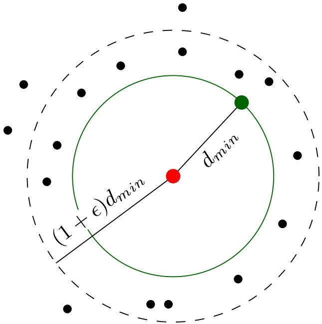
No difference (contrast) between nearest and farthest neighbors

A points nearest neighbor loses meaning



Instability of Nearest neighbor

A nearest neighbor query is ϵ-unstable ($\epsilon > 0$), if the distance from *q* and most other points are at most $(1+\epsilon)$ times the distance from *q* to its 1NN



We show that as dimensionality increases the probability of all nearest neighbors queries becoming unstable increase (distance concentration)

Distance Concentration

Another facet of curse of dimensionality is the phenomenon of distance concentration

Assume points in $\mathbb{R}^m$ and ℓ_2 distance measure

- As m increases, almost all pairs of points have their ℓ_2 distances
  - similar to distance of other pairs and
  - and very high
- normalized distance is close to 1 (both high and similar are encompassed)

We demonstrate it by observing distribution of pairwise distances for n points in $\mathbb{R}^m$ (again real-life datasets are not random...)

Distance Concentration

Another facet of curse of dimensionality is the phenomenon of distance concentration

All pairwise distances are very high

Consequences:

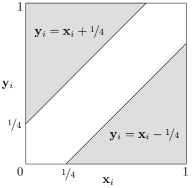
- Distance measure loses its meaning
- We discussed it earlier that proximity measure is the building block of data analytics, when it becomes meaningless the building collapses
- Nearest neighbor is as good as farthest neighbor
  - e.g. in such cases very hard to build clusters
  - no justification to group a pair of points and not another

Distance Concentration: Analytical Bounds

- Generate a set $\mathcal{X}$ of n points at random in $[0,1]^m$
- Maximum possible distance b/w a pair $\mathbf{x},\mathbf{y} \in \mathcal{X}$ is $d(\mathbf{x},\mathbf{y}) \leq \sqrt{m}$
- Consider the squared-ℓ_2 distance (for convenience)
- $d^2(\mathbf{x},\mathbf{y}) \; := \; \|\mathbf{x}-\mathbf{y}\|^2 \; \leq \; m$

Distance Concentration: Analytical Bounds

Generate a set $\mathcal{X}$ of n points at random in $[0,1]^m$



For a fixed coordinate $i < m$, $Pr[|\mathbf{x}_i - \mathbf{y}_i| \geq 1/4] > 1/2$

Let $V_i = \begin{cases} 1 & \text{if } |\mathbf{x}_i - \mathbf{y}_i| \geq 1/4 \\ 0 & \text{else} \end{cases}$ $\triangleright$ Indicator if coordinate difference is big

Let $V = \sum_{i=1}^{m} V_i = \left|\{i : |\mathbf{x}_i - \mathbf{y}_i| \geq 1/4\}\right|$

$E(V) \geq m/2$ $\triangleright$ linearity of expectation

On average at least half coordinates differences are $\geq 1/4$ ('big difference')

Distance Concentration: Analytical Bounds

Theorem (Chernoff Bound (tail inequality))

Let $V = V_1 + V_2 + \ldots + V_m$ be the sum of m independent Bernoulli random variables and let $E(V) = \mu$. The (loose) Chernoff bounds are:

- $Pr\big(V \ \geq\ (1+\delta)\,\mu\big) \ \leq\ e^{-\delta^2\mu/3}$ *for* $0 < \delta < 1$
- $Pr\big(V \ \geq\ (1+\delta)\,\mu\big) \ \leq\ e^{-\delta\mu/3}$ *for* $\delta > 1$
- $Pr\big(V \ \leq\ (1-\delta)\,\mu\big) \ \leq\ e^{-\delta^2\mu/2}$ *for* $0 < \delta < 1$

For fixed **x**, **y** $\left[\, V \geq \frac{m}{4} \implies \|\mathbf{x} - \mathbf{y}\|^2 \ \geq\ \frac{m}{64} \,\right]$ w.p $\geq 1 - e^{-\frac{m}{16}}$ $\triangleright\ \delta = \frac{1}{2}$

From this using union bound we get the following result

If $m = \Omega(\log n)$, then w.h.p for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ we have $d^2(\mathbf{x}, \mathbf{y}) \ \geq\ \frac{m}{64}$

This means all pairs are far (dist $\geq \sqrt{m}/8$)

Distance Concentration: Simulation

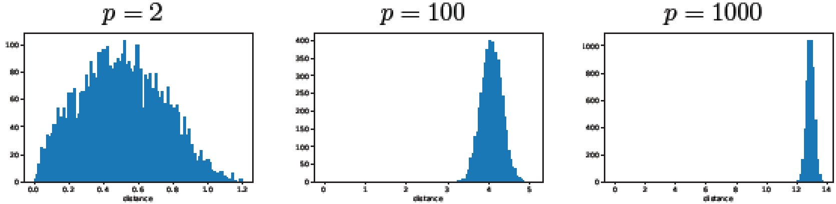


Figure: Histograms of pairwise-distances between $n = 100$ points sampled uniformly in the hypercube $[0, 1]^p$

Julie Delon @ Uni. Paris Descartes

Angle Concentration

- In large dimensions (at least for random points) the distance measure (at least ℓ_2 distance) is more or less meaningless
- Can we use cosine distance?
- The same concentration phenomenon is observed for pairwise angles

- Max num of pairwise orthogonal vectors ($\mathbf{x} \cdot \mathbf{y} = 0$, $\theta_{x,y} = 90^\circ$) in $\mathbb{R}^2$ is 2
- Max num of pairwise orthogonal vectors in $\mathbb{R}^3$ is 3
- Max number of pairwise almost orthogonal vectors in $\mathbb{R}^m$ ($\mathbf{x} \cdot \mathbf{y} \leq \epsilon$, $\theta_{x,y} = 90^\circ \pm \epsilon$) is $e^{\Omega(m)}$

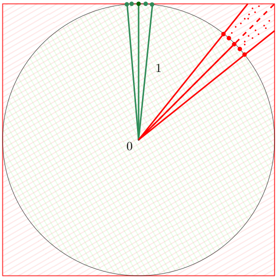
Angle Concentration: Random Direction

Generating a random direction in $\mathbb{R}^m$

- Equivalently a random unit vector in $\mathbb{R}^m$
- We will need it in subsequent sessions
- It is not a straight-forward task in higher dimensions

An immediate way to pick a random unit vector:

choose a random point in $\mathbf{v} \in [-1, 1]^m$ and normalize it as $\hat{\mathbf{v}} = \mathbf{v}/\|\mathbf{v}\|$



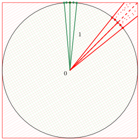
The red points have significantly high probability of begin chosen compared to the green points

Clearly the distribution is skewed towards the diagonal directions

Angle Concentration: Random Direction

Generating a random direction in $\mathbb{R}^m$

- choose a random point in $\mathbf{v} \in [-1, 1]^m$
- normalize it as $\hat{\mathbf{v}} = {}^{\mathbf{v}}/_{\|\mathbf{v}\|}$
- distribution skewed towards diagonal directions



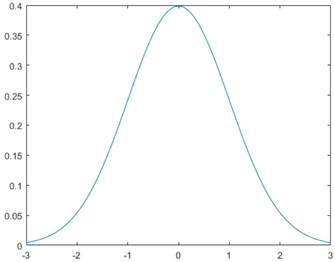
The red points have significantly high probability of begin chosen compared to the green points

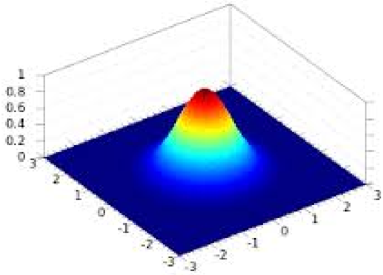
A quick fix, due to Marsaglia & Zaman

- Generate $\mathbf{v} \in [-1, 1]^m$
- If $\mathbf{v}$ is outside the unit hypersphere $(v_1^2 + v_2^2 + \dots v_m^2 > 1)$ discard it
- Normalize any non-discarded $\mathbf{v}$
  - we get a point on the surface of the unit-ball equally likely
- Computationally expansive $\triangleright$ diminishing volume of unit ball
- Just in 2d choose a random number in $[0, \pi]$ and make a unit vector

Angle Concentration: Random Direction

Generating a random direction in $\mathbb{R}^m$

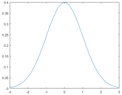


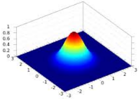


- Use spherical symmetry of the standard normal distribution
- Pick each coordinate $\mathbf{v}_i$ independently from $\mathcal{N}(0, 1)$ and normalize $\mathbf{v}$
- Known to be uniformly distributed over the surface of the unit m-ball

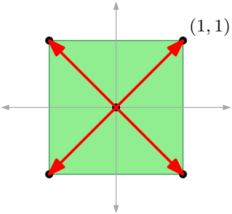
Angle Concentration: Approximate Random Direction

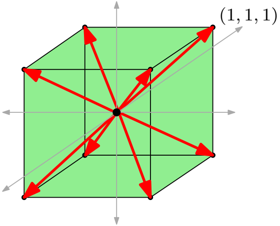
Generating a random direction in $\mathbb{R}^m$





- Approximately generate unit directions
  - generate directions towards corners of the m-cubes $[-1, 1]^m$
- For $m \gg 1$, these 2^m directions approximately cover surface of m-ball
- Achlioptas (2003), Database-friendly random projections: ...





Angle Concentration: Analytical Bounds

Generate a set $\mathcal{X}$ of n vectors in $[-1,1]^m$ ▷ and normalize them

$\mathbf{x}$ and $\mathbf{y}$ are orthogonal if $\cos\theta_{\mathbf{x},\mathbf{y}} = \langle \mathbf{x}, \mathbf{y} \rangle = \sum_i \mathbf{x}_i \mathbf{y}_i \sim 0$

For a fixed $\mathbf{x}$, let $V_i = \mathbf{x}_i \mathbf{y}_i$ and let $V = \sum_{i=1}^{m} V_i = \cos\theta_{\mathbf{x},\mathbf{y}}$

$$\frac{-\mathbf{x}_i}{m} \le V_i \le \frac{\mathbf{x}_i}{m} \qquad \text{and} \qquad E\left(V_i\right) = 0$$

On average the vector $\mathbf{x}$ is orthogonal to any vector $\mathbf{y}$

Angle Concentration: Analytical Bounds

Theorem (Hoeffding's Inequality)

If X_i's are random variables bounded by the interval $[a_i, b_i]$. Let $S = \sum_{i=1}^{m} X_i$. Then

$$Pr(|S - E[S] \geq t) \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^{m}(b_i - a_i)^2}\right)$$

Using this we get that

$$Pr(V \geq \epsilon) \leq 2\exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^{m}(2\mathbf{x}_i/m)^2}\right) = 2e^{-\epsilon^2 n/2}$$

From this using union bound we get the following result

If $m = \Omega(\log n)$, then w.h.p for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ we have $\cos\theta_{\mathbf{x},\mathbf{y}} \leq \epsilon$

This means all pairs are almost orthogonal (angle $\leq \arccos(\epsilon)$)