NAVAL
POSTGRADUATE
SCHOOL

OS4118

Statistical and Machine Learning

Inter-point Distances

Prof. Sam Buttrey

Fall AY 2020

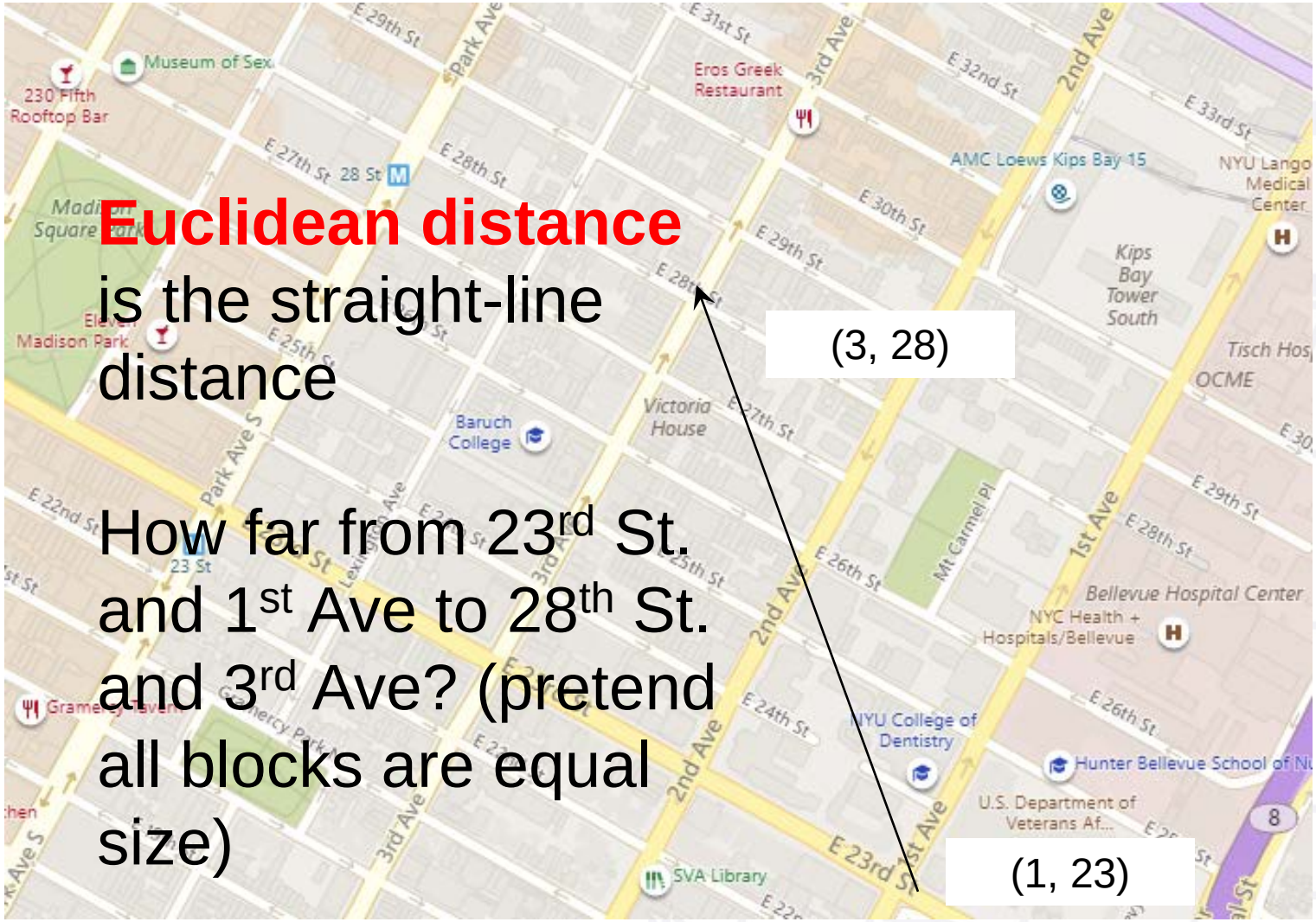
The Nation's Premiere Defense Research University

Monterey, California
WWW.NPS.EDU

Inter-point Distances

- Many statistics techniques (like clustering, to come) rely on a measure of **distance** (or ***dissimilarity***) between two points, between a point and a cluster, and between two clusters
- **Concerns: How do we…**
  1. Evaluate the contribution of a variable to the clustering (selection, weighting)?
  2. Account for correlation (or duplication) among variables?
  3. How do we incorporate categorical variables?

Euclidean Distance



Euclidean distance is the straight-line distance

How far from 23rd St. and 1st Ave to 28th St. and 3rd Ave? (pretend all blocks are equal size)

Source: Bing Maps

Euclidean Distance

Answer:

$\sqrt{[(1 - 3)^2 + (23 - 28)^2]}$

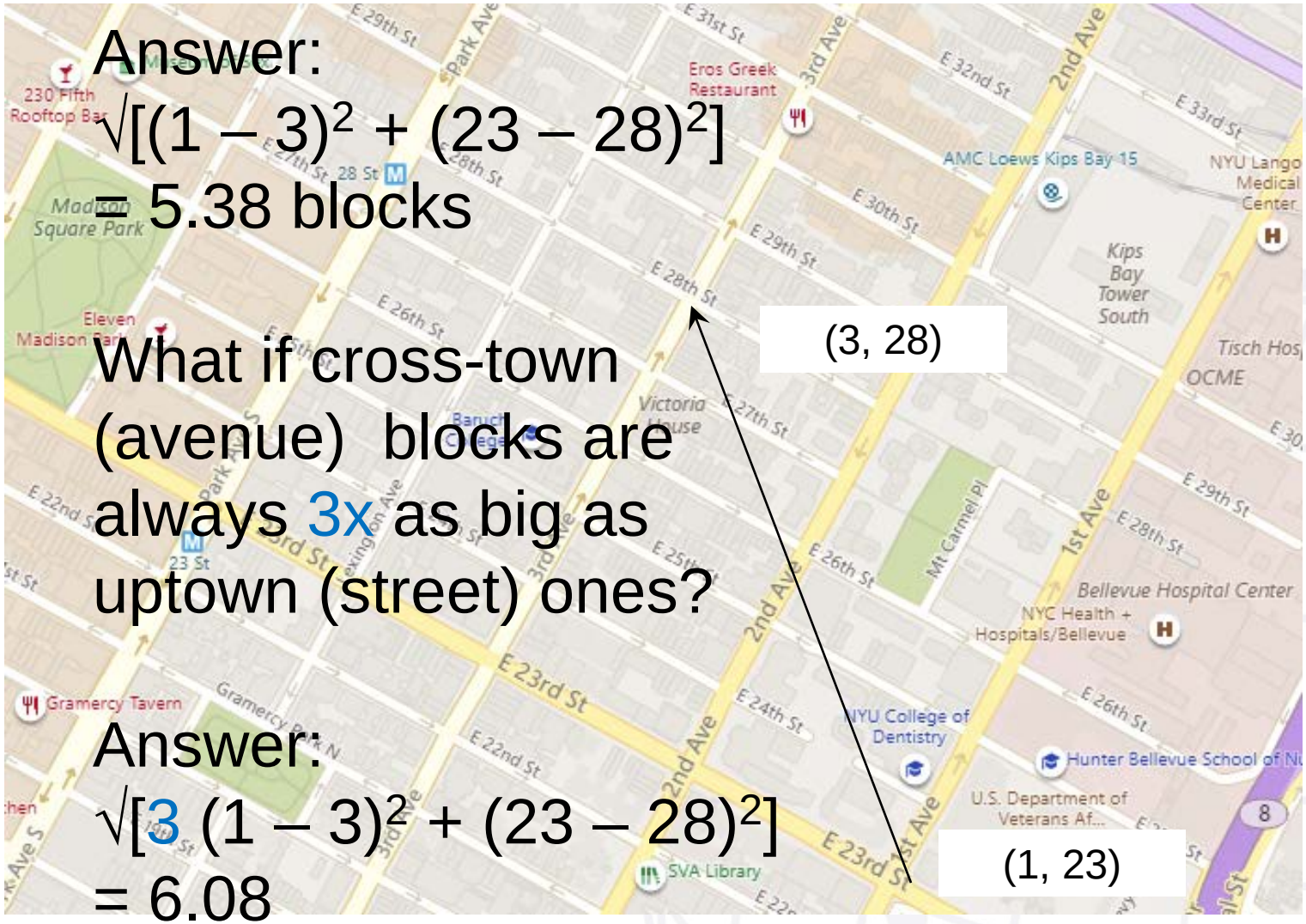
= 5.38 blocks

What if cross-town (avenue) blocks are always 3x as big as uptown (street) ones?

Answer:

$\sqrt{[3\ (1 - 3)^2 + (23 - 28)^2]}$

= 6.08



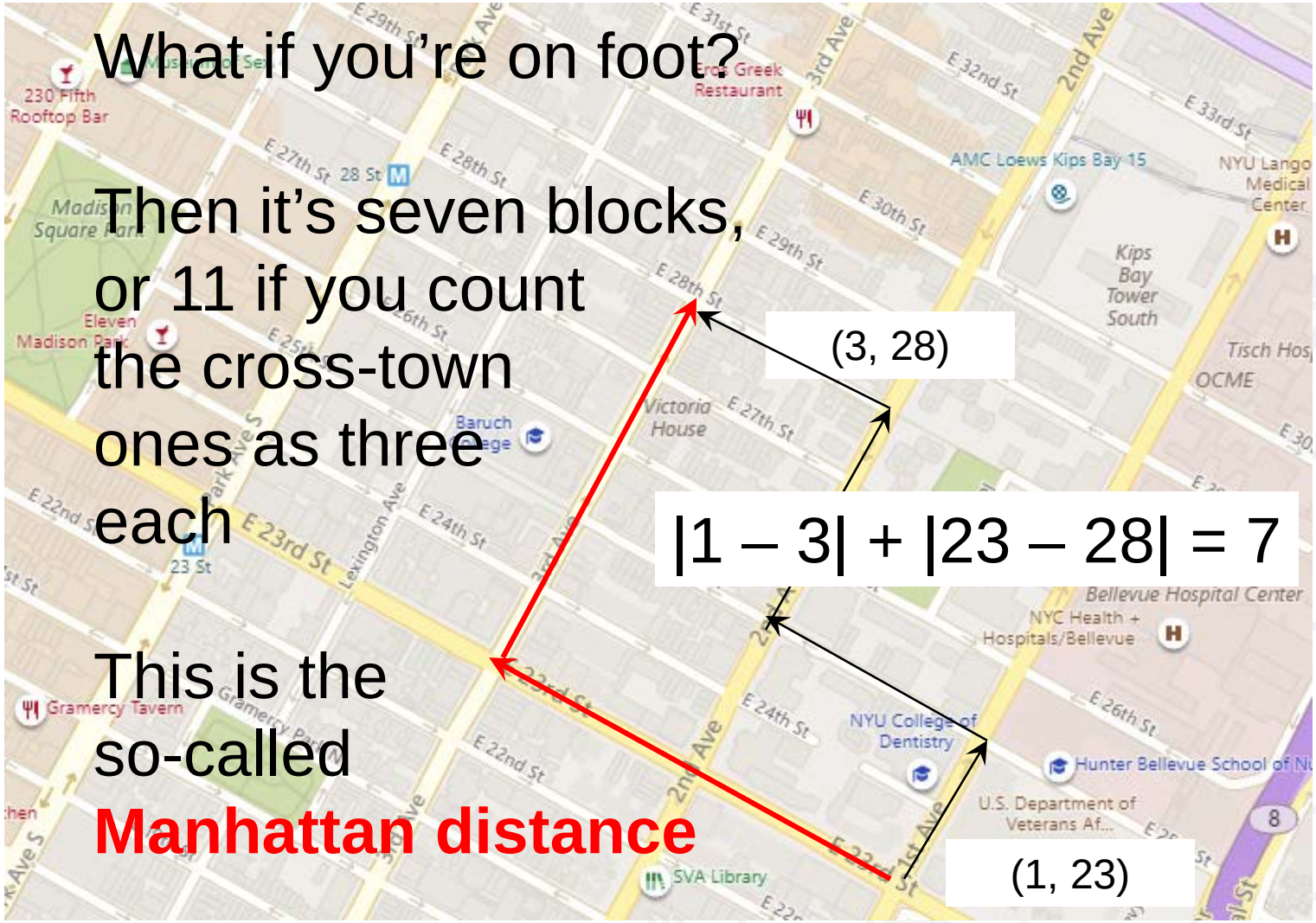
Source: Bing Maps

Manhattan Distance

What if you're on foot?

Then it's seven blocks, or 11 if you count the cross-town ones as three each

This is the so-called **Manhattan distance**



Source: Bing Maps

Distances

- Most techniques measure distance between two observations x_1, x_2 with:

$$d(x_1, x_2) = \sqrt{\sum_j \left(x_{1j} - x_{2j}\right)^2} \text{ (Euclidean) or}$$

$$d(x_1, x_2) = \sqrt{\sum_j w_j \left(x_{1j} - x_{2j}\right)^2} \text{ (wtd. Euc.)}$$

  - Weights w_j are 1, or sd (x_j), or range (x_j)
    - But are these good choices? Scale is not the same as "importance"
  - Correlation among X's usually ignored
  - Still needs modification for categorical data

Distances

- The weighted **Manhattan** distance is one alternative:

$$d(x_1, x_2) = \sum_j w_j \, |x_{1j} - x_{2j}|$$

More generally we could use the **Minkowski** distance for some value p:

$$d(x_1, x_2) = [\sum_j w_j \, |x_{1j} - x_{2j}|^p]^{(1/p)}$$

  - Again, choice of weights w_j are 1, or sd (x_j), or range (x_j)
  - Correlation among X's usually ignored
  - Still needs modification for categorical data

Mahalanobis Distance

- Suppose your data vectors x1, x2 come from a distribution with cov. matrix $\boldsymbol{\Sigma}$
- Then a "natural" way to account for correlation among observations is to weight by the inverse of that matrix:

$$d(x_1, x_2) = \sqrt{(\boldsymbol{x_1} - \boldsymbol{x_2})^{\mathrm{T}} \boldsymbol{\Sigma^{-1}} (\boldsymbol{x_1} - \boldsymbol{x_2})}$$

- This **Mahalanobis** distance generalizes the Euclidean…but the $\boldsymbol{\Sigma}$ matrix, $n \times n$, is unknown and hard to estimate

Distance Measure

- R's `daisy() {cluster}` computes inter-point distances (replaces `dist()`)
- Scale, choice of metric **can** matter
- If all variables numeric, choose "euclidean" or "manhattan"
- We can scale columns differently, but correlation among columns ignored
- Otherwise daisy uses **Gower** distance

Gower Distance

- If some columns are not numeric, the "dissimilarity" between numeric X_{ik} and X_{jk} scaled to $|X_{ik} - X_{jk}|$ / range (X_k)
  - (What happens when one entry in X_k has an outlier – like Age = 999?)
- For binary variables the usual dissimilarity is 0 if $X_{ij} = X_{ik}$, 1 if not
  - What if 1's are very rare (e.g. speaks Esperanto, attended Sorbonne)?
  - Asymmetric binary

Gower Distance

- Our observations are vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$
- The dist $d_{ij,k}$ between $\mathbf{x}_i$ and $\boldsymbol{x}_j$ on var. k is:
  - For categorical k, 0 if $x_{ik} = x_{jk}$, otherwise 1
  - For numeric k, $|x_{ik} - x_{jk}|$ / (range of column k)
- The overall distance d_{ij} is a weighted sum of these: $d_{ij} = \frac{\sum_{i=1}^{p} \partial_{ij,k}\, d_{ij,k}}{\sum_{i=1}^{p} \partial_{ij,k}}$
- Weights $\partial_{ij,k}$ are 1 except when one $\boldsymbol{x}$ is missing, or both 0 and $\boldsymbol{x}$ asymm.binary
- Case weights are also possible

Thoughts on Gower

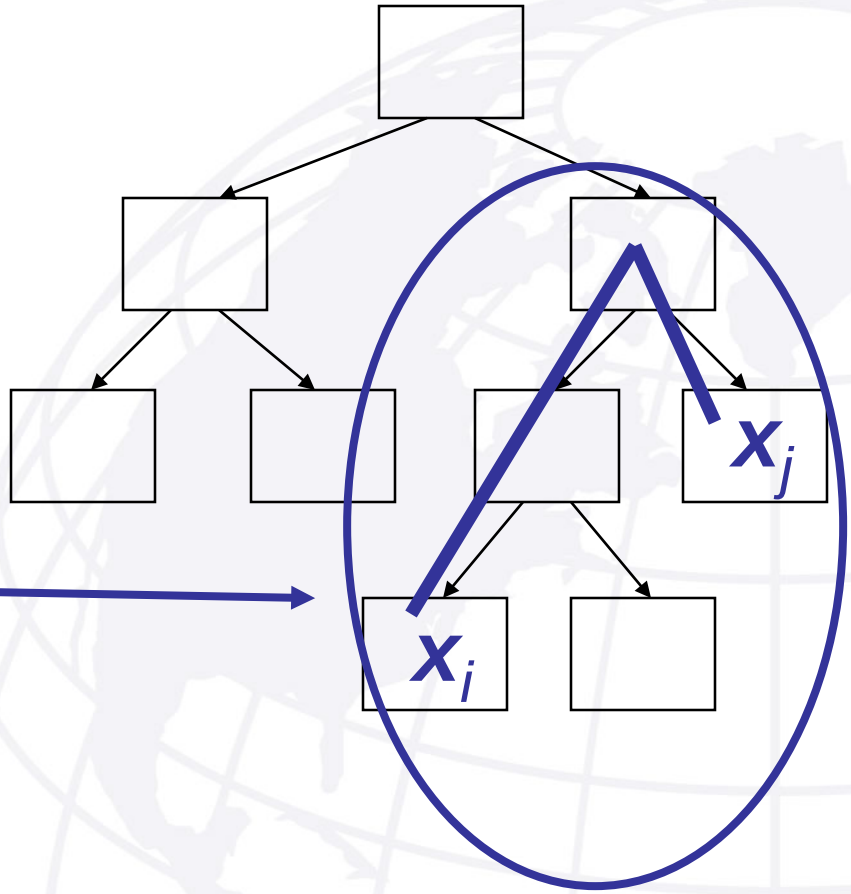
- Natural adjustment for missing values
  - Euclidean dist: inflate by √[ncol(X)/#non-NA]
- All these choices can matter!
- `daisy()` computes all the pairwise distances up front
- There are $n(n-1)/2$ of these, which causes trouble in really big data
- Things are different in high dimensions – our intuition is not very good here
- Dimensionality reduction is always good!

treeDist

- "Two observations are alike if they tend to fall in the same leaves of trees"
  - But trees require a response variable
- The treeDist distance computes *p* trees, with each column in turn as response
- Trees are pruned – some may be discarded
- The distance between points *i* and *j* on tree *t* is $d_1^t(i,j) = 0$ if *i* and *j* land in the same leaf, otherwise 1
- d_2 is like d_1 but with tree "quality" weights

Enhancements

- **Local** quality
- If two obs. are in different leaves, compute the change in deviance associated with the smallest sub-tree containing both



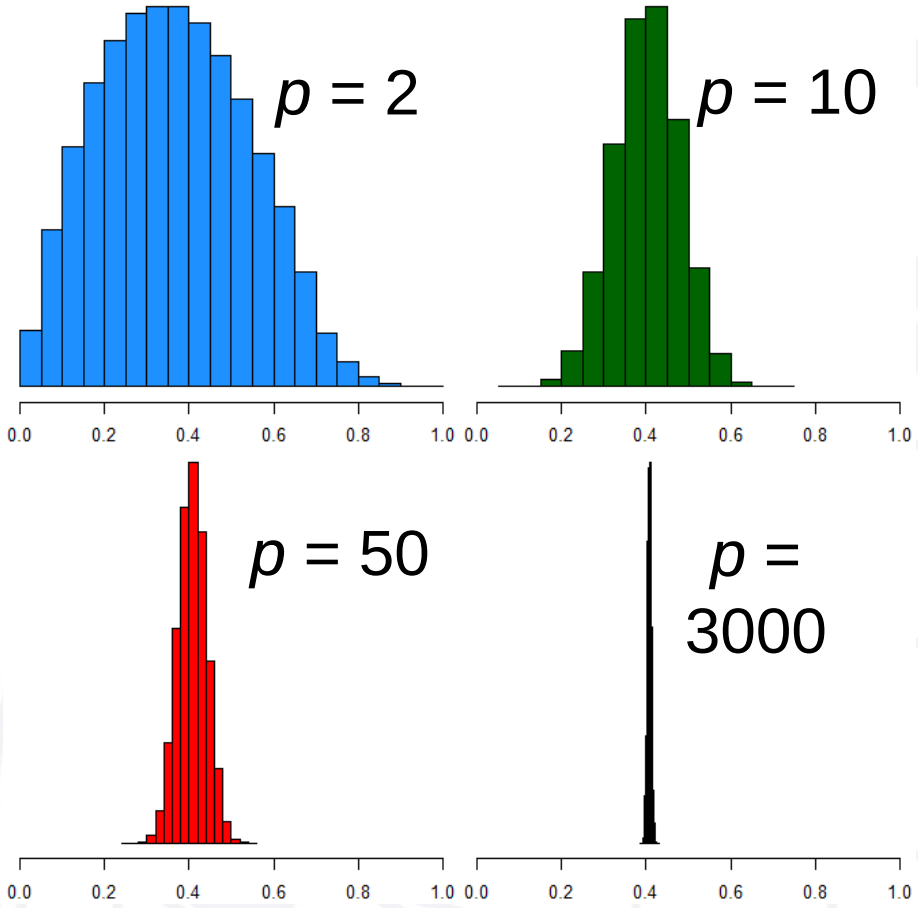
- Scale the (i, j) distance "accordingly" → d_3
- d_4: this "local" measure, weighted by tree quality
- The treeDist is unlike other measures in that it's learned from the data

Distances in Big Data Sets

- Instead of computing all $\binom{n}{2}$ pairwise distances, we can compute the set of pairwise **leaf distances** for each tree, which is much smaller
- We can also generate a new **numeric** data matrix whose inter-point distances "mirror" the inter-leaf ones
  - This data can be clustered or visualized in lower-d space via, e.g., multi-dim scaling
  - B. and W., *R Journal* 7/2, Dec. 2015
  - treeClust library at CRAN respository

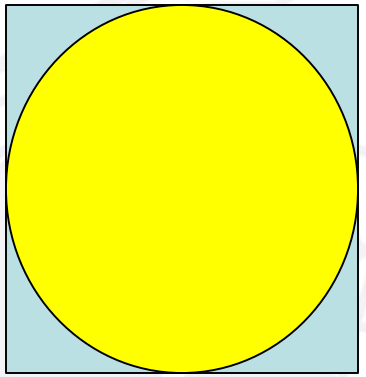
Digression: High-Dimensional Data

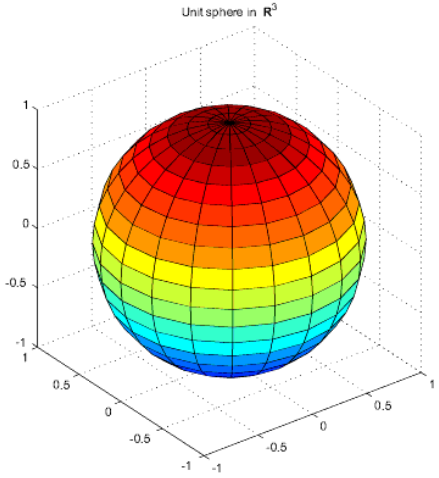
- High-dimensional data is just *different*
- Here are the pairwise distances among 1,000 points in *p* dimensions where each component is indep. U(–.5, +.5), scaled to (0, 1)
- In high dimensions, everything is "equally far away"
- Hopefully our data lies in a lower-dimensional subspace



Digression Part 2

- What proportion of the unit square is occupied by the unit circle?
- A: $\pi(1)^2 / 2^2 = \pi/4 = 0.79$
- What proportion of the unit cube is occupied by the unit sphere? A: $(4/3)\ \pi(1)^3 / 2^3 = \pi/6 = 0.52$





- How about in 30 dimensions? A: ≈ Zero!

Source: Matlab via http://www.ic.unicamp.br

Shameless Plug

- Remember random forests?
- "Proximity" measured by number of times two observations fell in the same leaf
  - But every tree has the same response variable
- The `treeClust()` dissimilarity of Buttrey and Whitaker (2015) creates (0 or) one tree from each response variable
  - Some trees are pruned to the root, dropped
  - Inherits tree missing value, outlier etc. handling
- Often performs well

Dimensionality Reduction

- **How** can we reduce dimensionality while preserving as much info as possible?
  - Principal Components, based on variance
  - "Projection Pursuit" is the name of techniques that use other criteria of "interestingness," like "as non-Gaussian as possible"
  - Sammon mapping: preserve the *ranking* of inter-point distances as much as possible
  - t-SNE seems pretty successful
- **Why**?
  - For plotting, visualization

NAVAL
POSTGRADUATE
SCHOOL

OS4118

Statistical and Machine Learning

Similar Items part 2 (Mostly a Digression)

Prof. Sam Buttrey

Spring 2019

The Nation's Premiere Defense Research University

Monterey, California
WWW.NPS.EDU

Finding Similar Items

- Goal: locate neighbors in scalable way
  - i.e. without computing $n(n-1)/2$ distances
  - Tasks
    - Find exact duplicates
    - Find nearest neighbor – or the k nearest
    - Find all, or at least one, neighbors within distance r
    - Find distribution of distances to k^{th} nearest
    - Find average number of points within r…

- Lescovec, Rajmaran, & Ullman, “Mining Massive Datasets,” On-line Textbook (2014)

Applications

- Numeric vectors:
  - Clustering
  - Multidimensional Scaling (more to follow)
    - These often require numeric variables – like most of the examples we've looked at
    - Need to handle mixed variables (e.g. Gower)
- Text documents
  - Plagiarism, mirror pages, same source articles
- Sets
  - Recommender systems

Distances

- Numeric vectors
  - **Euclidean** and **Manhattan** distances…
  - We know how to incorporate categoricals, asymmetric binaries…
  - …though scaling/weighting still needs to be considered…
  - …as does correlation – perhaps a simple sum of column distances is insufficient
  - `dist()` and `daisy()` in R

Distances (cont'd)

- Numeric vectors (cont'd)
  - **Cosine** similarity
    - $c(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T\mathbf{x}_2 / [\ ||\mathbf{x}_1||\ ||\mathbf{x}_2||]$
    - Often we take distance as 1 – similarity
  - Proportional to the "dot product" between the two vectors, but insensitive to scale
  - Recall $\mathbf{x}_1^T\mathbf{x}_2 = ||\mathbf{x}_1||\ ||\mathbf{x}_2||\ \cos(\theta)$, so the similarity is the cosine of the angle between the two vectors
  - $\cos(\theta) = 0 \Rightarrow \mathbf{x}_1, \mathbf{x}_2$ are parallel; $\cos(\theta) = 1 \Rightarrow \mathbf{x}_1, \mathbf{x}_2$ are perpendicular
  - E.g. in text mining, to compare two docs

Cosine Similarity Example

- Example: documents counts by word

	Doc 1	Doc 2	Doc 3
Gambia	15	0	0
baseball	0	23	12
Senegal	8	0	0
cotton	5	2	0
Oakland	0	0	9
peanuts	11	2	7
exports	0	0	1
shortstop	0	8	6
market	3	2	3

- $c(d_1, d_2) = 0.07$; $c(d_1, d_3) = 0.23$; $c(d_2, d_3) = 0.78$

Distances (cont'd)

- Numeric (usually binary) vectors
  - **Hamming** distance: the number of components on which the two differ
- E.g. $h(\texttt{001011}, \texttt{010011}) = 2$
- Extended to character sequences, possibly of different lengths:
  - **Levenshtein** (or "edit") distance: the number of one-character inserts, deletions, or changes necessary to turn one string into another
- E.g. L("button", "builtin") = 3

Distances (cont'd)

- Sets:
  - **Jaccard** distance, $d(A, B) = 1 - |A \cap B| / |A \cup B|$
  - The part on the right of the minus sign is "the fraction of all the items in both sets that appear in the intersection" – that is, set similarity
  - $d(A, B)$ measures how dissimilar two sets are
- Probability Distributions:
  - The distance from a dist'n p(x) to q(x) is often measured by the **Kullback-Leibler divergence**

$$\int_x p(x) \log \frac{p(x)}{q(x)} dx$$

Multidimensional Scaling

- Multidimensional scaling (MDS) is the general term for "compressing" a set of high-dimensional inter-point distances into a lower dimension (usually, 2 or 3)
- Find the low-d configuration whose inter-point distances are as similar as possible to the originals
  - So not the same as PCA, where we try to find new coordinates to capture the variance
  - Minimze **stress** between old and new

MDS, cont'd

- If the original distances are d_{ij}, and the lower-d ones are d^*_{ij}, then we might take

$$\text{Stress} = \sqrt{\frac{\sum\left(d^*_{ij} - d_{ij}\right)^2}{\sum d^2_{ij}}} \text{ or } \left(\frac{\sum\left|d^*_{ij} - d_{ij}\right|^p}{\sum d^p_{ij}}\right)$$

- MDS solutions will normally be insensitive to shift, rotation and reflection
- As with PCA we might need to **scale** data
- **Classical** MDA: minimize stress
  - There's a closed form solution for this

MDS, cont'd

- In **Metric MDS**, we minimize the stress based on a non-linear transformation of the distances

$$\text{Stress} = \left(\frac{\sum \left| d_{ij}^{*} - d_{ij} \right|^{p}}{\sum d_{ij}^{p}} \right)$$

with, e.g., $d_{ij}^{*} = a + b \log (d_{ij})$

- Some applications in, e.g. psychology
- Lots of alternate formulations of d_{ij}^{*} available

Non-Metric MDS

- In **Non-metric** MDS we try to respect the **ordering** of distances rather than their numeric values
- In "stochastic neighbor embedding" (SNE) we model the probabilities that observation *i* would pick *j* as a neighbor, and try to match those in the lower-d space…
  - Cost function from Kullback-Leibler
- …Leading to the widely used modification called **t-SNE**

MDS in R

- R has `cmdscale()` for classical MDS built in
- Metric MDS comes from packages, among them `smacof`
- Non-metric MDS from (among others) `ismMDS` and `sammon` in MASS library
- t-SNE from `Rtsne`
  - Best with smaller sample sizes
  - Remove duplicates

MDS Example

- **Splice data**
  - Primate spice-junction gene sequences
  - Each observation has a class EI, IE or n
  - Each observation has 60 A, C, T, G values
    - Except, inevitably, data prep required
  - Goal: Compute inter-point distances (excluding classes), map in 2 or 3 dimensions...
  - …with colors given by “real” class to try to see if the classes look different

NAVAL
POSTGRADUATE
SCHOOL

OS4118

Statistical and Machine Learning

Hashing (A Digression)

Prof. Sam Buttrey

Spring 2019

The Nation's Premiere Defense Research University

Monterey, California
WWW.NPS.EDU

Scalability

- We need algorithms to operate without computing all $n(n-1)/2$ distances
- For some applications it will be enough to identify a small number of candidate pairs, distances for which can then be computed explicitly

Hash

- A **hash** is a function (or verb, or noun) that describes converting the contents of an item (vector, document, video, set…) to one of a much smaller number of possibilities (e.g. an integer, a 32-byte string)
- The “hash table” is a related storage and look-up scheme
- Example: Dewey Decimal System

Hashing

- Goals of hashing:

1. Hash value should be easy to compute
2. The same source should produce the same hash value, but sources different by even a little should produce different values (*)
3. The hash should be **one-way**; that is, given a hash value, you should not be able to construct a string that also produces that hash value

Hashing

- The set of items with a particular hash value is called a **bucket**
  - So we hope each bucket has 0 or 1 items
  - A “collision” is when two different items end up in the same bucket
  - If we can’t have a unique hash output from each item, we at least want a uniform distribution of items over buckets

Hashing for Duplicates

- Imagine searching for duplicate documents
- Each doc has an ID and some contents
- 1.) Hash every document's contents; build list of (bucket, ID)
- 2.) If no bucket has two members, there are no duplicates
  - And we never had to compare two docs!
- 3.) Otherwise, examine all buckets with > 1 members for collisions/duplicates

Hashing for Data Integrity

- R packages carry MD5 signatures
- If your downloaded package produces the same signature as the one reported, you can be pretty sure you got the exact set of bits they sent
  - Or a collision of really tiny probability
  - If your MD5 value differs, your download is wrong, but you have no idea how

Bash example

- echo "The quick brown fox" | md5sum
- echo "The quick brown f0x" | md5sum
- Other algorithms (sha1sum, sha256sum) available
- For data integrity, these are all fine
- For crypto-strength, use SHA-2 or SHA-3 algorithms; md5 and sha1 have been “broken”

Digression: Public-Key Encryption

- A good hash is one-way: non-invertible
- A code or cipher needs to be reversible to recover the message
  - The “key” might be symmetric – same key encrypts and decrypts – in which case it must be protected
  - Asymmetric keys let recipient make the encoding key public while keeping the decoding key private: **public key encryption**
  - Asymmetric encryption is slow (by a factor of 1,000 compared to conventional?)

Digression: PGP

- Pretty Good Privacy (PGP) uses a random, symmetric **session key**
  - Session key must be kept secret, so…
- The message is encrypted with the session key; the session key is then encrypted with the receiver's public key
  - A hash of the message might be encrypted with the sender's private key, as well, and sent along as authentication

Digression: PGP (cont'd)

- Recipient decodes session key with receiver's private key, then decodes message using the session key
  - If she decodes the hash with the sender's public key she can compare the sender's hash with one she computes herself
  - If they match, there's a **digital signature** that can't be repudiated

Hashing for Crypto

- The CS and Crypto communities love hashing
  - They hate collisions
  - Tiny changes in the message need to produce big changes in the hash (*)
  - It should be difficult to generate a message with a particular hash – since collisions are rare, you would be “inverting” the hash and maybe recovering the original message

Locality Sensitive Hashing

- **LSH** takes a different approach
- Items that are “close together” should land in the same bucket with high probability
  - We welcome collisions, and want small changes in input to have small changes, or no change, in hash value
  - Anti-spam example: replacing “Rolex” with “R0lex” should have no effect w/high prob.

Families of Hash Functions

- Consider a family of functions, each function $f_i()$ producing a different hash
  - Ex: for Hamming distance, f_1 might be "value of bit *i* "
- We can **AND** or **OR** these functions together to try to control error rates
  - For Hamming, pick 40 of these, say
  - False positives: items that hash together but aren't alike (here, match at only 40 points)

LSH

- LSH measures have been implemented for Euclidean, Hamming, Jaccard and other distances
- Ex.: fingerprint matching[1] (Hamming)
- Imagine each fingerprint represented in a 1000-pixel image
- Each pixel may have a “minutia”

1 Lescovec, Rajmaran, & Ullman, “Mining Massive Datasets,” On-line Textbook (2014)

Fingerprints

- Problems:
  - 1.) Is this print in our database? (“one-many”)
  - 2.) Do any pairs in our database come from the same individual? (“many-many”)
- Assume that overall Pr (minutia) = **.20**
- Pr ($m_{ij \cdot k}$) = Prob. of a minutia at pixel *ij* of image *k*)
- Assume that, given two different images of the same finger, Pr ($m_{ij \cdot 2} \mid m_{ij \cdot 1}$) = .80
- For different fingers, Pr ($m_{ij \cdot 2} \mid m_{ij \cdot 1}$) = **.20**

Hash functions

- Pick three points at random
- Define the function $f()$:
  = 1 if all three have minutia, else = 0
- For two random fingerprints A and B,
  $\Pr(f(A) = 1, f(B) = 1) = .2^3.2^3 = .000064$
- For two images from the same finger,
  $\Pr(f(A) = 1, f(B) = 1) = .2^3.8^3 = .004096$
- What if we use lots of f () functions?
  – AND reduces false positives, OR reduces false negatives – usual trade-off

Combining $f()$s

- Create 1024 separate $f()$ functions, each referring to three random pixels
- What's the prob. that two fingerprints from the same finger match on **at least one** [this is OR] of the $f()$ functions?
  - A: $1 - (1 - .004096)^{1024} = .985$
- What's that prob. for two random prints?
  - A: $1 - (1 - .000064)^{1024} = .063$
- 1.4% false neg, 6.3% false pos

LSH

- The hash is easy to compute, even for a large database of prints, and the “at least one” part is easy to evaluate
- We can do even better than this in the fingerprints example (using AND)
- LSH works best when the overall level of similarity is not too high; different hashing approaches have been proposed for the case of lots of similarity

Ensembles

- This acts much the same way as bagging and other ensemble methods on classification and regression
- Combining lots of simple models (hash functions) can be more effective than creating one complex one
- Readily parallelizable; database can be distributed