**22.11.2020**

Discussed so far on Distance and Similarity:

Types of attributes (Nominal, Binary, Ordinal, Numeric)

Similarity = 1- Dissimilarity or Distance

* Started with a Data Matrix (objects or instances as rows and attributes or features as columns)
* Converted to a Dissimilarity Matrix (object x object matrix for each feature)
* Generation of Composite Dissimilarity Matrix taking all features together)

Obj\_id test-1 test-2 test-3

Nominal Ordinal Numeric

1 A excellent 45

2 B fair 22

3 C good 64

4 A excellent 28

Composite d(i,j) = ∑ δfij. dfij / ∑ δfij

f=1-p f=1-p

Dissimilarity or Distance Matrix

Nominal: 1 2 3 4 | Ordinal: 1 2 3 4

1 0 | 1 0

2 1 0 | 2 1.0 0

3 1 1 0 | 3 0.5 0.5 0

4 0 1 1 0 | 4 0 1.0 0.5 0

Numeric: |Xif – Xjf| / max – min

1 2 3 4

1 0

2 0.55 0

3 0.45 1.0 0

4 0.40 0.14 0.86 0

Composite: 1 2 3 4

1 0

2 0.85 0

3 0.65 0.83 0

4 0.13 0.71 0.79 0

Composite d(i,j) = ∑ δfij. dfij / ∑ δfij

f=1-p f=1-p

d(3,1) = (1\*1 +1\*0.5 + 1\*0.45) / 3 = 0.65

Work done so far:

* Pair wise comparison
* Individual value wise comparison

Work to be done:

* Set or vector wise comparison
* Comparison of non-numeric data
* Comparison of large set of data without or avoiding pair wise comparison.

Generalized distance measure can be classified as Euclidian and Non- Euclidian.

Usual Euclidian distance between two points: (x1-x2)2 + (y1-y2)2

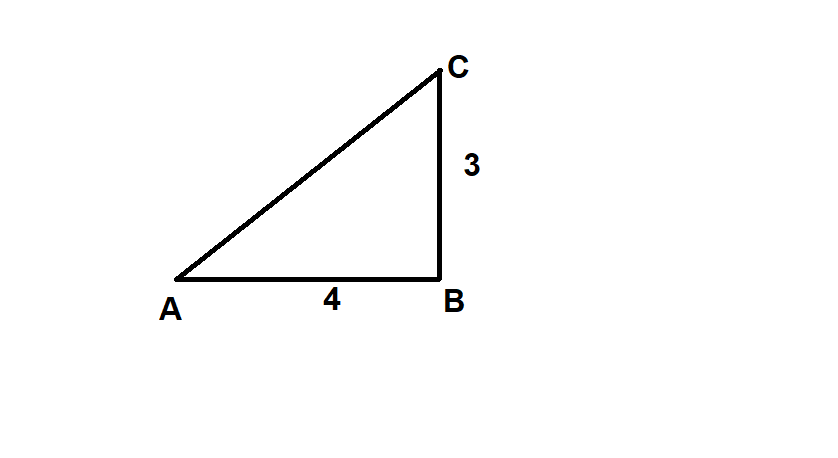
Distance between two vectors of n real numbers: (x1,x2,..xn) and (y1,y2,..yn) is ∑ (xi –yi)2

i=1 to n

This is a two dimensional distance called L2 norm. There can also be L1 norm.

x1 x2 x3

L1 distances: |x1-x2|, |x2-x3| and |x1-x3| = |x1-x2| + |x2-x3|



AC in L2 norm = (AB2 + BC2) =5

AC in L1 norm = AB + BC = 4 + 3 = 7 (Manhatten distance)

Minkowski distance: (∑ (xi –yi)h )1/h Lh norm

i=1 to n

L∞ norm when h ∞, also called Lmax, Supremum distance, Chebyshev distance or Uniform norm.

Weighted Euclidian distance = ∑ wi.(xi –yi)2

i=1 to n

Jaccard Distance or Jaccard Similarity:

V1 {1,2,3,4} V2 {1,3,5}

Jsim = (V1 in V2) / (V1 un V2) = 2/5

Jdis = 1-2/5 = 3/5

Jaccard Bag Distance or Jaccard Bag Similarity:

V1 {a,b} V2 {a,b,c}

Jsim = (V1 in V2) / (V1 un V2) = 2/3

Jdis = 1-2/3 = 1/3

V1 {a,a,a,b} V2 {a,a,b,b,c}

(V1 in V2) = {a,a,b} & (V1 un V2) = {a,a,a,b,a,a,b,b,c}

Jsim = (V1 in V2) / (V1 un V2) = 3/9 = 1/3

Jdis = 1-1/3 = 2/3

Hamming Distance: Binary Data

10101 11110

HD = 3

**23.11.2020**

Edit Distance (String distance)

X = abcde Y = acfdeg (Convert X to Y)

In X,

1. Delete b
2. Add f after c
3. Add g after e d(x,y) =3

Largest common subsequence = acde =4

*Assertion:*

Edit distance = (x+y) – 2\*LCS = 5+6 -2x4 =11-8=3

*Justification:*

1. If (X in Y) = φ then Eddis = X+Y
2. If (X in Y) = LCS then Eddis = (X-LCS) + (Y-LCS)

*Example:*

X = abc Y = def Eddis = 3+3 = 6

X = abg Y = deg LCS = g Eddis = (3-1) + (3-1) = 4

***Is Edit Distance a valid distance measure?***

*Mutation* and *Transposition* are counted as 1 (one).

**Reference:**

Section 3.5.1 of “Mining of Massive datasets” – A Rajaraman, J Leskovec and J D Ullman, Stanford University.

Chapter 2 of “Data Mining Concepts & Techniques” – J Han, M Kamber and J Pei

Cosine Distance: distance between two vectors.

V1 = (5,0,3,0,2,0,0,2,0,0)

V2 = (3,0,2,0,1,1,0,1,0,1)

Cos (V1,V2) = V1.V2 / ||v1||. ||v2|| = 25 / 42. 17 = 0.94 almost 10

(5x3+0x0+3x2+……) = 25

||v1|| = 52 + 32 +….. =42, ||v2|| = 17

Utility Matrix

Items m1 m2 m3 m4 m5 m6

A 3 2 5 - - -

B 2 3 3 3 4 -

C 5 4 4 - 5 -

Cos (A,B) = 0.167 Cos (A,C) = 0.662

Cosine similarity.

**02.12.2020**

Recommender or Recommendation System:

Why we need recommendation?

* Pre-internet era provided limited set of products at retailers.
* Online sources like Amazon, Flipcart can have many more varieties.
* Selecting one out of so many may need recommendations.

Areas where recommendation may be needed against user options:

Books or articles, products, manufacturers, movie, tourist places, hotels etc.

Types of recommendations:

* Hand curated: list of favorites, list of essential items. There is no user input.
* Simple aggregates: Top 10, most popular, recently uploaded
* **Tailored to individual users**: Amazon, Netflix etc. Depending on the taste of an individual.

Towards a formal model:

C = set of customers & s = set of items

Utility function U: CxS -> R (set of ratings). Gives rise to a Utility matrix that maps each customer against each item to a rating. It may be a star rating like 1\* to 5\* or a score bounded between [0,1]. Rating is always an ordered set.

Utility matrix

Items m1 m2 m3 m4 m5

A 3 2 5 - -

B 2 3 3 3 4

C 5 4 4 - 5

Utility matrix is usually sparse and a key problem of a recommendation system is to judge the missing values. Other problems are:

1. Gathering ratings (supposed to be known) –Explicit, Implicit (how to get low ratings).

Asking directly – explicit way.

Implicit way. Purchase figure – scalable. How to get low rating?

Mixture of explicit and implicit way, hybrid.

1. Extrapolating unknown ratings from known ratings (interested only in high ratings)
2. Evaluation of the recommendation system.

Most people have not rated most of the items.

New items have no ratings. |

New users have no history. | Cold start problem.

Three approaches to recommender systems:

1. Content based – from user’s past access history & content of items.
2. Collaborative filtering – recommended items based on similarity measures between users. Items preferred by similar users.
3. Latent factor based.

Content based system

Recommend items to X similar to items rated highly by X. Needs to create user and item profiles.

Recommend movies against same actors, director, genre.

Recommend books, articles having similar content.

Recommend people having many common friends.

Plan of Action:

For each user find the set of items liked by him/her.

For each item study its features and create a feature vector.

Generate a catalog (user x item and set of users x set of items).

Recommend from catalog.

How to generate a User profile:

Each item is a vector, take simple avarage (how many times an item has appeared), next weighted average (weights given by the user)

Let’s take a binary vector ( presence or absence of an actor)

Movies -> m1 m2 m3 m4 m5

Actor A 1 0 1 0 0 | A’s wt.=2/5

Actor B 1 1 0 1 0 | B’s wt.=3/5

Star Rating (1 to 5)

A’s rating by user X -3,5 | Mean rating = 3

B’s rating by user X -1,2,4 |

Diviation from Mean and normalized rating:

For A = 0,2 Profile wt = (0+2)/2=1

For B = -2,-1,1 Profile wt = (-2-1+1)/3=-2/3

Recommendation using user profile and item profile when they are high dimensional vectors. Distance -> Cos θ = x.i / (||x|| . ||i||)

Adv: No need of data from other users.

Can recommend to users with unique taste.

Can justify recommendations easily.

Disadv: Content based approach is hard as creation of Item profile is very hard (music, book, movies) etc.

Another problem is overspecialization – never recommends outside a user’s item profile, people may have multiple interest that cannot be exploited, unable to take into account judgement of other users. Cold start is a serious problem. Profiles cannot be built easily and made trustworthy.

Collaborative Filtering:

Study the users in the neighborhood and recommend from their ratings.

Utility matrix for Movie recommendation: (In a scale of 0 to 5)

Items m1 m2 m3 m4 m5 m6 m7

A 4 - - 5 1 - -

B 5 5 4 - - - -

C - - - 2 4 5 -

D - 3 - - - - 3

For each recommender there is a rating vector.

Need to generate a similarity metric sim(x,y)=?

Intuitively: sim(A,B) > sim(A,C)

Methods:

Jaccard Similarity = rA ∩ rB / rA U rB

sim(A,B)=1/5 and sim(A,C)=2/5 => sim(A,B) < sim(A,C)

Against intuition, why? Ignores rating values.

Utility matrix replacing unrated items by 0:

Items m1 m2 m3 m4 m5 m6 m7

A 4 0 0 5 1 0 0

B 5 5 4 0 0 0 0

C 0 0 0 2 4 5 0

D 0 3 0 0 0 0 3

By Cosine similarity:

sim(A,B)=0.38 and sim(A,C)=0.32 => sim(A,B) > sim(A,C)

Difference is very small, why? Missing values are replaced by worst rating.

Centered Cosine measure:

1. Find Row average.
2. Subtract avg from rating for modified rating.
3. Modified row sum is 0, making 0 as the centered rating.
4. Replace missing values by 0.

Items m1 m2 m3 m4 m5 m6 m7

A 2/3 0 0 5/3 -7/3 0 0

B 1/3 1/3 -2/3 0 0 0 0

C 0 0 0 -5/3 1/3 4/3 0

D 0 0? 0 0 0 0 0?

Sim(A,B)=0.09 > Sim(A,C)= -0.56

* Captures Intuition better
* Missing ratings are considered as average.
* Handles both “tough raters” and “easy raters”.

Pearson Correlation.

**Reference:**

Section 3.5.1 and Part of Section 9 from “Mining of Massive Datasets” – A Rajaraman, J Leskovec and J D Ullman, Stanford University.

Chapter 2 of “Data Mining Concepts & Techniques” – J Han, M Kamber and J Pei

**07.12.2020**

Rating Predictions:

Prediction to be made for user *x* against an item *i*.

Let *N* be the set of *k* users most similar to *x* who rated the item *i*.

*rxi* = *1/k* ( ∑*y ϵ N ryi*)

This rating does not consider similarity values of each *y* in *N*.

Modified rating (weighted average)

*rxi* = ( ∑*y ϵ N Sxy ryi*) / ( ∑*y ϵ N Sxy*) where, *Sxy* = sim (*x,y*)

Work done so far is the User-User collaborative filtering.

Similarly there can also be Item-Item collaborative filtering.

To find the rating of item *i* for user *x* from rating of other users on items similar to *i*.

Users

**1 2 3 4 5 6 7 8 9 10 11 12**

m1 1 3 **?** 5 5 4

m2 5 4 4 2 1 3

m3 2 4 1 2 3 4 3 5

m4 2 4 5 4 2

m5 4 3 4 2 2 5

m6 1 3 3 2 4

Find the rating of movie **m1** for the User **5**.

*Sm1,m2* = -0.18, *Sm1,m3* = 0.41, *Sm1,m4* = -0.10, *Sm1,m5* = -0.31, *Sm1,m6* = 0.59

m1 row avg=18/5 and m2 row avg=19/6. Two rows under centered Cosine are: -

Users

**1 2 3 4 5 6 7 8 9 10 11 12**

m1-13/5 0 -3/5 0 2.6 7/5 0 0 7/5 0 2/5 0

m2 0 0 11/6 5/6 0 0 5/6 0 0 -7/6 -13/6 -1/6

m3 2 4 1 2 3 4 3 5

m4 2 4 5 4 2

m5 4 3 4 2 2 5

m6 1 3 3 2 4

*rxi* = ( ∑ *j ϵ N Sij rxj*) / ( ∑ *j ϵ N Sij*)

* For item *i* find other similar items.
* Estimate rating of item *i* based on ratings of similar items.
* Use same similarity metrics and prediction functions as in user-user model.

*r5,m1* = (0.41\*2 + 0.59\*3) / (0.41 + 0.59) = 2.6

* Theoretically user-user and item-item are dual approaches and should have similar impact.
* In practice, item-item gives a better result in most usecases.
* Items belong to only a small set genres but a user may have varied tastes.
* Item similarity is more meaningful than user similarity.

**Implementation ?**

Finding *k* most similar users or items.

Simple measure gives O(n.|U|) where n is the number of users or items and |U| is the size of utility matrix.

Needs to be pre-computed.

**Methods:** Near neighbour search in high dimension, Clustering, Dimensionality reduction.

* So even pair wise comparison against a single reference entity may be computationally prohibitive.
* Problem of finding similarity among all pairs is even worse.
* Possible applications:
* Pages with similar words: classification by topics.
* Recommendation systems: Fans to movies and Movies to fans.
* Entity resolution.

**Application considered**:

Finding similar webpages (webpages having substantially similar texts).

First, convert informal documents to formal value sets – Shingling

Then, generate signatures of documents for comparison – Minhashing

Then finding similar documents – Locality Sensitive Hashing (LSH)

* Instead of comparing texts, compare sequence of characters and find such sequences in common
  + Finding mirror sites and search engine will connect to only one such site.
  + Identifying plagiarism even large quotations.
  + Grouping news items and clustering by “same story”.

Common method for such comparison is Jaccard Similarity =

rA ∩ rB / rA U rB

However it is important to find out how many such comparison to be made.

Three important methods –

Shingle: Converting documents to sets of character strings.

Minhashing: Convert large sets to short signatures maintaining the similarity.

Locality Sensitive Hashing (LSH): Comparing pairs of signatures which are likely to be similar.

However, these processes may give rise to both false positive and false negative results.

So the method is to create Shingles from documents, followed by creation of signatures by Minhashing and then selecting pairs to be compared by LSH and ultimately all pair-wise comparison – to be done in the main memory.

Generation of k-shingle (k-gram): Generation of sequence of k characters. Example: abcab -> 2- shingle -> {ab, bc, ca}

Representing a doc by a set of k-shingles.

Similar documents will have many shingles in common.

Changing a word will affect only k-shingles within a distance of k from the original word.

Reordering paragraphs affect only 2k-shingles at the paragraph boundaries.

Example: “The dog which chased the cat” and “The dog that chased the cat” -> compare with 3-shingles -> 7 shingles replaced by 6 shingles.

**12.12.2020**

**Shingling** creates sets of strings of length *k* that appear in a document.

**Minhashing** creates short signatures in the form of integer vectors that represent the sets and reflect their similarities. It takes much less space and can be done even in main memory.

**Locality Sensitive Hashing (LSH)** results in creating the candidate pairs that may be taken for pair wise comparison.

Result is taking much less space and time than of the order of n2.

For example: Possible set of 10-shingles –

(26C 10).10!, but all of them are not valid words. Can be hashed to 4 bytes called Tokens. Saves about 60% space to store actual shingle sets.

So a shingle set is actually a set of tokens with very small chance of conflict.

Now a document can be represented by a set of k-shingles or corresponding tokens.

So similarity between two documents can be measured by

Jaccard Similarity = d1 ∩ d2 / d1 U d2

Create a binary matrix. Columns are possible universal set of k-shingles and rows are the documents. If a k-shingle is present then the entry is 1 else 0. So similarity between two columns is the Jaccard Similarity of all 1,s. Typically the matrix is very sparse.

Between two columns, there can be 4 types of rows (11,01,10,00) or (a,b,c,d). So sim (c1,c2) = a/(a+b+c)

Shingles s1 s2 …….. Sn

Documents

D1 1 0 …………….. 1

D2

D3

D4

.

.

Dm

C1 C2

1. 1 a
2. 0 b
3. 1 c

0 0 d sim(c1,c2) = a / a+b+c -> Jaccard Similarity

**16.12.2020**

**Problem:** Matching words with documents for selecting high degree of similarities among documents for detailed analysis using Association Rule Mining techniques.

Words -> W1 W2 …….. Wm

Documents

D1 1 0 …………….. 1

D2

D3

.

.

.

Dn

Each document has presence or absence of a word in the matrix along with corresponding normalized word count.

**Find the set of documents that have many words in common with significant number of appearances of those words.**

1. 1-itemset counting.
2. Arrange the words in descending order of their frequencies.
3. Retain only words with high degree of appearance.
4. Result is - Smaller matrix (both rows and columns are reduced).
5. Computing TF for each word already selected.