CS660: Mining Massive Datasets
University at Albany—SUNY

April 28, 2016

1 Intro (Ch. 1)

1.1 Bonferoni's principle examples

"If your method of finding significant items returns significantly more items that you would expect in the actual population, you can assume most of the items you find with it are bogus."

"Finding terrorists":

- 10^9 people being tracked
- Examine hotel records for 1,000 days
- Each person stays in a hotel 1% of time (1 day out of 100)
- Hotels hold 100 people (so 10^5 hotels)

Detection: People visiting the same hotel on two different days are terrorists.

What is the probability of an event to occur at random?
P(two people decide to visit a hotel)=$10^{-2}10^{-2} = 10^{-4}$
P(two people visit the same hotel)=10^{-5}
P(same hotel on one given day)= $10^{-4}10^{-5} = 10^{-9}$
P(same hotels on two given days)= 10^{-18}

How many possible events?
(# pairs of people) x (# pairs of days) = $\binom{10^9}{2}\binom{10^3}{2} \approx \frac{(10^9)^2}{2}\frac{(10^3)^2}{2} =$
$5 \times 10^{17} \times 5 \times 10^5$

Expected number of events at random?
$10^{-18} \times 25 \times 10^{22} = 250000$
If number of actual terrorists is small, say 10s, it will be intractable to check 1/4 million people to find them. Also privacy?

"People cheating at a company":
We know that 5% of people at a company cheat on their spouses.
Detection: People who say they go out with colleagues once a weak are cheaters.

Assume that it is a social company and 20% answer the above criterion. At best 1/4 of our suspects are actual cheaters (True positives). Also it is likely that there will be FN - people who do not answer the criterion but ARE cheaters.

You will encounter similar challenges in DM often, where beyond an answer you would like to ask: How significant is this answer? Or alternatively: How likely is it to obtain it at random? A very important decision to make regarding such analysis is what is a reasonable model for "at random".

1.2 TF.IDF

The TF.IDF measure is used to weight the importance of individual terms t in a collection of documents $D = \{d_i\}$ to the topic of a specific document. The definition is as follows:

$$TF(t,d) = \frac{n_{td}}{\sum_{k \in T} n_{kd}},$$

where n_{td} is the number of occurrences of term t in document d and T is the set of all terms. Note, that the normalization may be also by $\max_{k \in T} n_{kd}$ instead of $\sum_{k \in T} n_{kd}$.

$$IDF(t) = log\frac{|D|}{|\{d|t \in d\}|},$$

where $\{d|t \in d\}$ is the set of documents containing at least one occurrence of t. Informally, high TF.IDF is associated to terms that are frequent in a document but are not found in many other documents.

2 MapReduce (Ch. 2)

2.1 Matrix Multiplication in MR

We would like to compute: $P = M \times N$

$$p_{ik} = \sum_j m_{ij} n_{jk}$$

We assume a sparse representation of M and N and the result P.

MR Task 1:
Map:
In: $(i, j, m_{ij}, M), (j, k, n_{jk}, M)$
Out: $(j, (M, i, m_{ij})), (j, (N, k, n_{jk}))$
Reduce:
For all values for a given j, create pairs such that one comes from M and the other from N
Out: $((i, k), (m_{ij} * n_{jk}))$

MR Task 2:
Map:
In: $((i, k), (m_{ij} * n_{jk}))$
Out: $((i, k), (m_{ij} * n_{jk}))$ [No Change]
Reduce:
Sum values for each (i, k)
Out: $(i, k), (\sum_j m_{ij} * n_{jk}))$

Q: How to do it with a single MR task? Idea: m_{ij} contributes to many sums: $\forall k$, same for n_{jk}

Single MR Task for Matrix Multiplication:
Map:
In: $(i, j, m_{ij}, M), (j, k, n_{jk}, M)$
Out: $((i, k)(M, j, m_{ij})), \forall k = 1..cols(N)$,
$((i, k)(N, j, n_{jk})), \forall i = 1..rows(M)$,

Reduce:
In: $(i, k) -> (M, j, m_{ij}) .... (N, j, n_{jk})))$. Sort by j, compute products $m_{ij} n_{jk}$. Maintain the sum of all products.
Out: $((i, k), (\sum_j m_{ij} * n_{jk}))$

Q: Problems? Size of the reducer might be too big, need to write/read from disk. and do external sort in the reducer. Also need to know the dimensions of matrices in advance.

2.2 TF.IDF in MapReduce

$$TF.IDF(t,d) = \frac{n_{td}}{\sum_{k \in T} n_{kd}} \times log \frac{|D|}{|\{d|t \in d\}|}$$

Need to compute each of the quantities:
- n_{td} - frequency of t in d
- $n_d = \sum_{k \in T} n_{kd}$ - sum (or max) of all frequencies in a document d
- $n_t = |\{d|t \in d\}|$ - # docs containing t
- $|D|$ - number of documents

Task n_{td}:
Map:
In: (d, content from d) (could be chunked)
Out: $((d,t),1)$ (Combiner?)
Reduce:
Sum counts for key (d,t).
Out: $((d,t),n_{td})$

Task $\sum_{k \in T} n_{kd}$ (or max):
Map:
In: $((d,t),n_{td})$
Out: $(d,(t,n_{td}))$
Reduce:
Sum n_{td} for key d (or max).
Out: $((d,t),(n_{td},n_d))$

Task $n_t = |\{d|t \in d\}|$ and output results
Map:
In: $((d,t),(n_{td},n_d))$
Out: $t,(d,n_{td},n_d,1)$
Reduce:
Sum 1s for every t to get n_t. In a second iteration over all $t,(d,n_{td},n_d,1)$ output the results
Out: $((d,t),(n_{td}/n_d \times log(|D|/n_t))$, assumes $|D|$ is known, or s simple MR to compute it

3 Finding Similar Items (Ch. 3)

3.1 Proof that Jaccard distance is a metric

Jaccard distance is defined as:

$$JD(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

It is easy to verify:

- Symmetry: $JD(A,B) = JD(B,A)$
- Non-negativity: $J(A,B) \geq 0$, and
- Identity of indiscernables: $J(A,B) = 0$ iff $A = B$

What is not as trivial is to show triangular inequality. Namely,

$$JD(A,B) \leq JD(A,C) + JD(C,B)$$

Theorem 1. *The triangular inequality holds for the Jaccard distance.*

Proof. We consider 3 sets S_1, S_2, S_3 drawn as Venn diagrams in the following figure[1] :

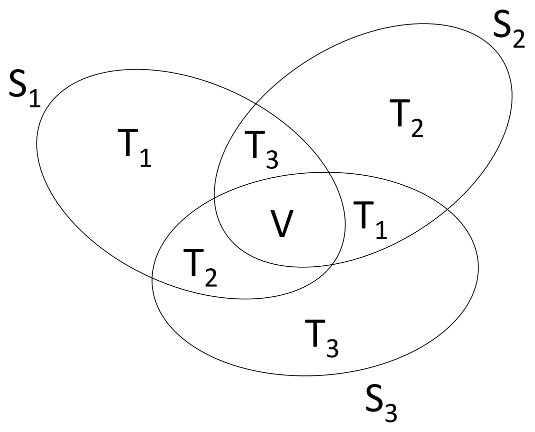


Figure 1: Three sets and some intersections.

Let $U = S_1 \cup S_2 \cup S_3$, $V = S_1 \cap S_2 \cap S_3$ and the sets T_i be defined according to the Venn diagram. Based on the definitions of T_i, it follows that:

$$\frac{|T_1| + |T_2| + |T_3|}{|U|} = 1 - \frac{|V|}{|U|} \tag{1}$$

Next, one can show that the Jaccard distance for an arbitrary pair is bounded from above by the quantity in equation Eq. 1:

$$1 - \frac{|V|}{|U|} \geq 1 - \frac{|S_i \cap S_j|}{|S_i \cup S_j|} = JD(S_i, S_j) \tag{2}$$

The above inequality follows from (i) $|S_i \cap S_j| \geq |V|$ and (ii) $|S_i \cup S_j| \leq |U|$, hence $\frac{|V|}{|U|} \leq \frac{|S_i \cap S_j|}{|S_i \cup S_j|}$.

The Jaccard distance for an arbitrary pair is bounded from below based on the following:

$$JD(S_i, S_j) = \frac{|S_i \cup S_j| - |S_i \cap S_j|}{|S_i \cup S_j|} = \frac{|T_i| + |T_j|}{|S_i \cup S_j|} \geq \frac{|T_i| + |T_j|}{|U|} \tag{3}$$

Applying Eq. 3 for the sum $JD(S_i, S_j) + JD(S_j, S_k)$ we get the triangular inequality as follows:

$$JD(S_i, S_j) + JD(S_j, S_k) \geq \frac{|T_i| + 2|T_j| + |T_k|}{|U|} \geq \frac{|T_i| + |T_j| + |T_k|}{|U|} \geq JD(S_i, S_k),$$

where the last inequality follows from Eqs. 1,2.

□

3.2 Expected Jaccard Smilarity of random sets

Is a $JS = 0.8$ high? It depends. We always need to ask what JS do we expect at random.

Assume that we are comparing subsets of size m, where the elements are from a universe $U, |U| = n$, so $0 \leq m \leq n$. We would like to know what is $E[JS(S,T)] =?$

Random Model: Assume that we generate the subsets at random by drawing from U without replacement to form m-element sets.

[1]Based on G. Gilbert, "Distance Between Sets" Nature 239, 1972

To compute the expectation, we need to consider all possible values of the intersection and the union of the sets and the probabilities of drawing those.

Assume that the intersection $|S \cap T| = k$, then $J_k = k/(2m - k)$. Why?

What is the probability of observing an intersection of size k, $P(\cap_k)$?
- Number of ways to choose first set: $\binom{n}{m}$

- Number of possible subsets of size k for the intersection: $\binom{m}{k}$

- Number of possible ways to choose the non-intersection items in the second set: $\binom{n-m}{m-k}$.

Hence, we get the following:

$$P(\cap_k) = \frac{\binom{n}{m}\binom{m}{k}\binom{n-m}{m-k}}{\binom{n}{m}\binom{n}{m}} = \frac{\binom{m}{k}\binom{n-m}{m-k}}{\binom{n}{m}}$$

The expectation, can be computed as follows:

$$E[JS(S,T)] = \sum_{k=0..m} J_k P(\cap_k) = \frac{k}{2m-k}\frac{\binom{m}{k}\binom{n-m}{m-k}}{\binom{n}{m}}.$$

Note: $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ is the binomial coefficient.

3.3 Minhashing

Minhashing allows us to represent sets as uniform controlled-size summaries (signatures). Let S be a set of elements from a universe $U, |U| = n$ and $\pi : \{1...n\} \to \{1...n\}$ be a random permutation of the indices of all possible elements. The minhash value of S is defined as: $h(S) = argmin_{S_i \in S}\pi(i)$, i.e. i is the index of the first element that is part of the set S according to the order of indices in π.

Minhashing is similarity-preserving in that if $JS(S_1, S_2)$ is high, then with high probability $h(S_1) = h(S_2)$ and if $JS(S_1, S_2)$ is low, then with high probability $h(S_1) \neq h(S_2)$.

Theorem 2. *Let $h()$ be a minhash function, then $P(h(S_1) = h(S_2)) = JS(S1, S2)$.*

Proof. Consider the bitmap representation of S_1 and S_2, where the indices of included elements are set to 1. There are three types of indices:
X: indices in which both S_1 and S_2 are both 1
Y: indices in which only one of S_1 or S_2 is 1
Z: indices in which both S_1 and S_2 are zero

$$JS(S_1, S_2) = \frac{|X|}{|X| + |Y|}$$

Since the indices are permuted randomly each index has an equal chance to be first in the permutation. Since indices of type Z do not affect the hash value in order to compute $P(h(S_1) = h(S_2))$ we need to consider what is the chance that an X index is firs in the permutation among all X and Y indices. Formally, $P(h(S_1) = h(S_2)) = \frac{|X|}{|X|+|Y|}$. □

The signature/summary of a set, based on k independent hash functions is the vector $[h_1(S), ...h_k(S)]$. If we define the similarity of two signatures $s([h_1(S_1), ...h_k(S_1)], [h_1(S_2), ...h_k(S_2)])$ as the fraction of dimensions in which the signatures agree, the above theorem gives us the following:

$$JS(S_1, S_2) = E[s([h_1(S_1), ...h_k(S_1)], [h_1(S_2), ...h_k(S_2)])],$$

i.e. the expected signature similarity equals JS.

3.4 Locality sensitive functions

Decision functions that render two items either a candidate (close) pair or not. Often the function is a hashing function (e.g. minhash) and $h(x) = h(y)$ results in declaring (x, y) a candidate pair.

Desirable properties:

1. More likely to make close pairs candidates.
2. Functions in the family are statistically independent
3. Efficient - fast decision on candidate/non-candidate
4. Combinable- reduce FP and FN when combined, e.g. banding

Definition 1. *Let $d_1 < d_2$ be two distance values for a distance measure d. A family of functions F is (d_1, d_2, p_1, p_2)-sensitive if $\forall f \in F$:*

1. *if $d(x, y) \le d_1$ then $P(f(x) = f(y)) \ge p_1$*
2. *if $d(x, y) \ge d_2$ then $P(f(x) = f(y)) \le p_2$*

Let F be a (d_1, d_2, p_1, p_2)-sensitive family of functions.

AND-combination: We can combine members by a logical "AND" r times to obtain F^r_{AND}. The decision logic is (x, y) is a candidate if $f_1(x) = f_1(y) \cap ... f_r(x) = f_r(y)$. Then F^r_{AND} is (d_1, d_2, p_1^r, p_2^r)-sensitive (i.e. both p_1, p_2 decrease)

OR-combination: We can combine members by a logical "AND" b times to obtain F^b_{OR}. The decision logic is (x, y) is a candidate if $f_1(x) = f_1(y) \cup ... f_b(x) = f_b(y)$. Then F^b_{OR} is $(d_1, d_2, 1 - (1 - p_1)^b, 1 - (1 - p_2)^b)$-sensitive (i.e. both p_1, p_2 decrease)

Cascading: For example, r times AND combined b times by OR. $(F^r_{AND})^b_{OR}$ is $(d_1, d_2, 1-(1-p_1^r)^b, 1-(1-p_2^r)^b)$-sensitive (Does it remind you of something?)

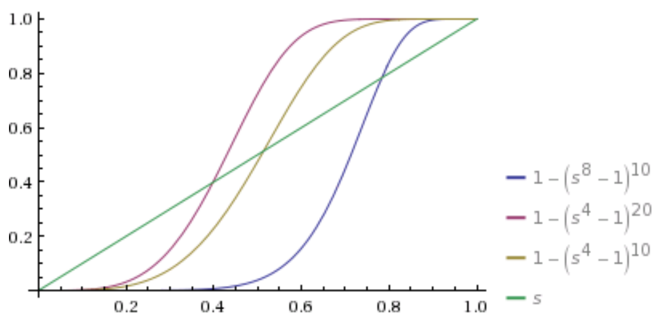


Figure 2: Effect of cascading.

Example: Consider a $(0.2, 0.8, 0.8, 0.2)$-sensitive family F (MinHash). Construct:

$F1$: 4-way AND and then 4-way OR on F to get:
$(0.2, 0.8, 1 - (1 - 0.8^4)^4, 1 - (1 - 0.2^4)^4) = (0.2, 0.8, 0.88, 0.006)$

$F2$: 4-way AND and then 4-way OR on $F1$ to get:
$(0.2, 0.8, 1 - (0.88^4)^4, 1 - (1 - 0.006^4)^4) = (0.2, 0.8, 0.97, 5 \times 10^-9)$

Data	**Distance**	**LS Family**	**Sensitivity**
Sets	$JD(X,Y) = 1 - \frac{\lvert X\cap Y\rvert}{\lvert X\cup Y\rvert} = 1 - s$	MinHash	$(1 - s_1, 1 - s_2, s_1, s_2)$
Sets	$JD(X,Y) = 1 - \frac{\lvert X\cap Y\rvert}{\lvert X\cup Y\rvert} = 1 - s$	MinHash+Banding r, b	$(1 - s_1, 1 - s_2, 1 - (1 - s_1^r)^b, 1 - (1 - s_2^r)^b)$
d-dim. vec.	Hamming: $\lvert\{i\lvert v_1(i) \neq v_2(i)\}\rvert$	Equal in a random i	$(d_1, d_2, 1 - d_1/d, 1 - (1 - d_2/d)$
d-dim. vec.	Cosine: 1- $cos(\theta_{v_1,v_2})$	$sign(< v, r >)$, r-rand. vec.	$(d_1, d_2, 1 - d_1, 1 - d_2)$

4 Stream processing Ch. 4

4.1 Cardinality estimation using FM-sketches

FM-sketches are named after the original authors Flajolet and Martin [1]. They are designed for counting distinct items in stream.

Let S be a multi-set of integers between 0 and N (those could simply be indices of items in a finite universe that may appear on the stream). We will denote the number of unique items observed as F, this is the number we would like to find. The assumption is that we do not have $O(N)$ space for a hash table to maintain whether each of the possible items is observed. Instead, we will be happy with an approximate answer $\tilde{F}$.

Let w be the minimum number of bits necessary to encode any of the integers in $[0, N]$, i.e. $w = logN$. Let also $h()$ be an ideal hash function mapping values in $k \in S$ uniformly at random to values between $[0, 2^w - 1]$, i.e. bit strings of length w.

Let z_k be the number of trailing zeroes in the hash value $h(k)$ of element k. For, example, for $h(k) = 000100$, $z_k = 2$ as there are 2 trailing zeroes. Let

$$Z = max_{k \in S} z_k,$$

which can be computed using only a single scan over the whole stream and a memory of $O(logN)$. The estimate produced by the FM-sketch is

$$\tilde{F} = 2^Z$$

.

We would like to get a guarantee on the quality of the estimate $\tilde{F}$ w.r.t. the actual F. We will prove that:

Theorem 3. *For every integer $c > 3$, $\frac{1}{c} \leq \frac{\tilde{F}}{F} \leq c$ with probability greater than $1 - 3/c$.*

While the probabilistic guarantee might not seem great $1 - 3/c$ it can be made arbitrarily close to 1 using multiple independent sketches and taking the median of their estimates. We will first show some intermediate results that will be later used in the proofs.

Lemma 1. *For all integers $r \in [0, w]$*

$$P(z_k \geq r) = \frac{1}{2^r}$$

.

Proof. $z_k \geq r$ means observing at least r trailing zeroes in $h(k)$. The bit strings that answer this criterion are of the between:

$$\underbrace{0...0}_{w-r}\underbrace{0...0}_{r}$$

and

$$\underbrace{1...1}_{w-r}\underbrace{0...0}_{r}$$

where the prefix can be set to any string and the suffix contains r zeroes. There are 2^{w-r} such strings and a total of 2^w possible unconstrained strings. Since $h(k)$ is a uniformly random function, the probability of k being hashed to a string with suffix of at least r zeroes is $2^{w-r}/2^w = 1/2^r$. □

For a fixed r, let us define an indicator function $x_k(r) = (z_k \geq r)$, i.e. $x_k(r) = 1$, when $z_k \geq r$ and 0 otherwise. Another way to state the lemma above is, $\forall k \in S \; P(x_k(r) = 1) = 1/2^r$.

Next, we derive the expectation and variance of the binary random variable $x_k(r)$:

$$E[x_k(r)] = 1(\frac{1}{2^r}) + 0(1 - \frac{1}{2^r}) = \frac{1}{2^r}$$

$$var[x_k(r)] = E[x_k^2(r)] - E^2[x_k(r)] = 1^2(\frac{1}{2^r}) + 0^2(1 - \frac{1}{2^r}) - (\frac{1}{2^r})^2 = \frac{1}{2^r}(1 - \frac{1}{2^r})$$

Let $X(r) = \sum_{k \in S} x_k(r)$, i.e. $X(r)$ is 0 if all $x_k(r) = 0$ and it is greater than or equal to 1 if at least one hash value has a longer trail of leading zeroes.

We will also define two special constants:

1. r_1 is the smallest r such that $2^r > cF$
2. r_2 is the smallest r such that $2^r \geq F/c$

Lemma 2. *If $X(r_1) = 0$ AND $X(r_2) \neq 0$, then the algorithm is correct, meaning $1/c \leq \frac{\tilde{F}}{F} \leq c$*

Proof. To show this equivalence we will look at each of the assertions:

If $X(r_1) = 0$, then no hash value had more than r_1 trailing zeroes, hence $\forall k z_k < r_1$, hence $Z < r_1$. Then we can bound our APX/OPT fraction as follows:

$$\frac{\tilde{F}}{F} = \frac{2^Z}{F} = \frac{2^{r_1-1-\epsilon}}{F} \leq cF/F,$$

Note that for the inequality we used the fact that Z is an integer and $Z < r$ and since r_1 is the smallest r, s.t. $2^r \geq F/c$, then for all r smaller the r_1, including Z the opposite inequality holds.

If $X(r_2) \neq 0$, then at least one hash value has at least r_2 trailing zeroes, hence $\exists k, z_k \geq r_2$, hence $Z \geq r_2$. Then we can bound our APX/OPT fraction as follows:

$$\frac{\tilde{F}}{F} = \frac{2^Z}{F} = \frac{2^{r_2+\epsilon}}{F} \geq \frac{2^{r_2+\epsilon}}{F} \geq F/cF = 1/c,$$

Again the last inequality is due to the fact that $r_2 \geq F/c$ and Z is at least that. smallest r, s.r. $2^r \geq F/c$, then for all r smaller the r_1, including Z. □

We will bound the probability of the algorithm not working correctly. i.e.

$$P[\neg(X(r_1) = 0 \vee X(r_2) \neq 0)] = P[X(r_1) \neq 0 \wedge X(r_2) = 0].$$

We will show that $P(X(r_1) \neq 0) < 1/c$ and that $P(X(r_2) = 0) < 2/c$ from which we will get that the algorithm is correct with probability greater than $1 - 3/c$.

Lemma 3. $P(X(r_1) \geq 1) < 1/c$

Proof.

$$E[X(r_1)] = \sum_{\text{distinct } k \in S} E[x_k(r_1)] = \sum_{\text{distinct } k \in S} \frac{1}{2^{r_1}} = \frac{F}{2^{r_1}} < \frac{1}{c},$$

The first equality follows from the fact that $X()$ is a sum of of i.r.v. The last inequality is due to the definition of r_1: it is the smallest r s.t. $2^r > cF$, and hence the inequality in the opposite direction as 2^{r_1} is in the denominator.

Next from Markov inequality we have that:
$P(X(r_1) \geq 1) \leq \frac{E[X(r_1)]}{1} < 1/c$ □

Lemma 4. $P(X(r_2) = 0) < 2/c$

Proof.

$$E[X(r_2)] = F/2^{r_2}$$

, due to the same reasoning as in the previous lemma proof.

$X(r_2)$ is the sum of i.r.v., hence its variance is the sum of the individual variances:

$$var[X(r_2)] = F\frac{1}{2^{r_2}}(1 - \frac{1}{2^{r_2}}) < \frac{F}{2^{r_2}}$$

$$P[X(r_2) = 0] \leq P[|X(r_2) - E[X(r_2)]| \geq E[X(r_2)]]$$

The above follows from a reasoning about the event in the first probability and how it is a subset of the events in which the expression in the second probability is satisfied. Hence the second probability is higher.

$$P[|X(r_2) - E[X(r_2)]| \geq E[X(r_2)]] \leq \frac{var[X(r_2)]}{E^2[X(r_2)]}$$

The above follows from Chebyshev bound. Substituting the inequality and equality we derived earlier respectively for the variance and expectation of $X(r_2)$, we obtain.

$$P[X(r_2) = 0] \leq \frac{var[X(r_2)]}{E^2[X(r_2)]} < \frac{F/2^{r_2}}{(F/2^{r_2})^2} = \frac{2^{r_2}}{F} < 2/c$$

To see the last inequality, consider the definiion of r_2: it is the smallest r s.t. $2^{r_2} \geq F/c$. This means for r smaller than r_2 the inequality will be reversed and specifically for $r_2 - 1$: $2^{r_2-1} < F/c$, and hence $2^{r_2} < 2F/c$ leading to the last inequality in the above.

□

5 Clustering Ch. 7

5.1 The curse of dimensionality

High-dimensional vectors are ubiquitous in data mining. In high dimensions we encounter the so-called *curse of dimensionality* which refers to the fact that algorithms are simply harder to design in high dimensions.

Let $x \in \mathbb{R}^n$ be an n-dimensional vector. Its L_2-norm is defined as $|x|_2 = (\sum_i x_i^2)^{1/2}$ and its L_1 norms as $|x|_1 = \sum_i |x_i|$. The Euclidean (L_2) distance of two vectors is $|x - y|_2$ and the Manhattan (L_1) distance is $|x - y|_2$.

The *n-cube* in $\mathbb{R}^n$ is the set of all points with coordinates between 0 and 1, i.e. $\{x | 0 \leq x_i \leq 1\}$. Its volume is 1.

The *unit ball* in $\mathbb{R}^n$ is $B_n : \{x | \sum x_i^2 \leq 1\}$ the set of vector tips, whose squared L_2 norm does not exceed 1. The volume of this ball is $\frac{\pi^{n/2}}{(n/2)!}$ (assume n is even), which using Big-Θ notation is $\frac{1}{n^{\Theta(n)}}$. An intuitive (approximate) way to reason about choosing a point on the surface of the ball is to choose a random point x, such that $x_i = \{-1, 1\}$ and normalizing it to get $\frac{1}{\sqrt{n}}x$.

A few facts and their proofs

1. The volume of B_n tends to 0 as the number of dimensions n tend to ∞.

This follows directly from the volume definition $\frac{\pi^{n/2}}{(n/2)!}$ or $\frac{1}{n^{\Theta(n)}}$ taking n to infinity. The unit cube's volume on the contrary goes to infinity with the number of dimensions. So if you imagine a hyper-sphere within a hyper-cube (even if not unit), as the number of dimensions increases, most points will be outside the sphere close to its boundary and hence will have similar distance to each other.

2. If you pick two vectors on the surface of B_n independently, then with probability $> 1 - 1/n$, the cos of the angle between them is:

$$|cos(\Theta_{x,y})| = O(\sqrt{\frac{logn}{n}}),$$

In other words the vectors are almost orthogonal.

Lemma 5. *Suppose a is a unit vector in $\mathbb{R}^n$, i.e. $|a|_2 = 1$. Let x be chosen from the surface of B_n by choosing each coordinate from $\{-1, 1\}$ independently and normalizing by a factor of $1/\sqrt{n}$. Note, that x is also a unit vector but from a restricted type. Denote by X the random variable $< a, x >= \sum_i a_i x_i = \frac{\sum_i a_i x_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i x_i^2}} = cos(a, x)$. Then:*

$$P(|X| > t) < e^{-nt^2}$$

Proof. Since the -1 and 1 are equally likely in the dimensions of x, $E[x_i] = 0$. We can then show that:

$$E[X] = E[\sum a_i x_i] = \sum a_i E[x_i] = 0$$

Similarly considering the fact that $E[x_i * x_j] = 0, i \neq j$ and $E[xi * xi] = 1$ we have:
$\sigma^2 = var[X] = E[X^2] - E[X] = E[(\sum_i a_i x_i)^2] = E[\sum_{i,j} a_i a_j x_i x_j] = \sum_{i,j} a_i a_j E[x_i x_j] = \sum_i a_i a_i E[x_i^2] = \frac{a_i^2}{n} = \frac{1}{n}$
Using Chernoff bound, we get:
$P(|X| > t) < e^{-t^2/\sigma^2} = e^{-t^2/(1/n)} = e^{-nt^2}$ □

We fixed a to a unit vector and showed something about the $cos(a, x)$. The ssame will hold for random vector constructed similar to x as they are a special case of unit vectors. We mentioned that we will approximately heuristically think about them as truly random vectors on the hyper-sphere.

As a direct corollary from the Lemma, if two vectors x, y are chosen independently:

$$P(|cos(\Theta_{x,y})| > \sqrt{\frac{-log(\epsilon)}{n}}) < e^{-n\sqrt{\frac{-log\epsilon}{n}}^2} = e^{log\epsilon} = \epsilon$$

In order to get claim 2. we set $\epsilon = 1/n$.

$$P\Big(|cos(\Theta_{x,y})| > \sqrt{\frac{logn}{n}}\Big) < 1/n,$$

hence

$$P\Big(|cos(\Theta_{x,y})| \leq \sqrt{\frac{logn}{n}}\Big) \geq 1 - 1/n$$

5.2 Spectral Clustering

Given a dataset of points $x_1 ... x_n$ and some notion of similarity $s_{ij} \geq 0$ the intuitive goal is to group points such that points within groups have high similarity and points across groups have low similarity.
Notation A natural way to model this situation is via a similarity graph $G(V, E, W)$: every vertex v_i represents a point, two vertices are connected by an edge in E if their similarity is high, and the weight on the edge $w_{ij} = s_{ij}$. G is undirected, i.e. its adjacency matrix is symmetric. There are different ways to set this up:

1. Pick a threshold ϵ and retain edges if the similarity exceeds the threshold, or alternatively if working with distances if the distance is smaller than ϵ. In the latter scenario, one needs to convert a distance to a similarity.
2. Retain the top k most similar neighbors for every vertex. This is referred to as kNN graph.
3. Complete similarity graph - keep all possible edges.

W is the symmetric adjacency matrix of G and its non-zero elements correspond to edges that we have retained. The degree of a node is the sum of the weights of its adjacent edges:

$$d_i = \sum_{j}^{n} w_{ij}.$$

The **degree matrix** D is defined as a diagonal matrix with node degrees on the diagonal $d_{ii} = d_i$ and 0 in all off-diagonal entries.

The **weigh between two sets of nodes** $A, B \subset V$ is defined as the sum of all weights across:

$$W(A, B) = \sum_{i \in A, j \in B} w_{ij}.$$

Two ways to measure the **size** of a subset A of nodes:

1. Number of nodes $|A|$
2. Volume $vol(A) = \sum_{i \in A} d_i$.

The intuitive objective of grouping similar objects, corresponds to a **partitioning** of the graph $\{A_1 ... A_k\}$, where partitions are mutually disjoint $A_i \cap A_j = \emptyset$ and cover all nodes $\bigcup_i A_i = V$.

Note, that spectral clustering is not necessarily applicable to data in the form of points in Euclidean space, or objects with a similarity function. One's objective could be partition a set of users in a social network in which case W will correspond to the friendship graph.

The weighted adjacency matrix W is symmetric $w_{ij} = w_{ji}$ and has non-negative entries $w_{ij} \geq 0$. An **eigen pair** of a symmetric square matrix is defined as:

$$Wv = \lambda v,$$

where v is called an **eigenvector** and λ the corresponding **eigenvalue**. Intuitively, v does not change its direction when linearly transformed by W. If W is a square matrix of size n, it has n eigenpairs, such that every pair of eigenvectors are orthogonal. We will arrange the eigenpairs in increasing order of their eigenvalues.
Graph Laplacians The main tools for spectral clustering.

The **Combinatorial Laplacian** L is defined as $L = D - W$. It has the following properties:

1. It is a quadratic form, i.e. captures square differences of adjacent nodes

$$f \in \mathbb{R}^n, f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

Proof.

$$\begin{aligned} f'Lf = f'Df - f'Wf = \sum_{i=1}^{n} f_i^2 d_i - \sum_{i,j=1}^{n} w_{ij} f_i f_j = \frac{1}{2}\Big(\sum_{i=1}^{n} f_i^2 d_i - 2\sum_{i,j=1}^{n} w_{ij} f_i f_j + \sum_{j=1}^{n} f_j^2 d_j\Big) \\ = \frac{1}{2}\Big(\sum_{i=1}^{n} f_i^2 (\sum_{j}^{n} w_{ij}) - 2\sum_{i,j=1}^{n} w_{ij} f_i f_j + \sum_{j=1}^{n} f_j^2 (\sum_{i}^{n} w_{ij})\Big) \\ = \frac{1}{2}\Big(\sum_{i,j=1}^{n} f_i^2 w_{ij} - 2\sum_{i,j=1}^{n} w_{ij} f_i f_j + \sum_{i,j=1}^{n} f_j^2 w_{ij}\Big) \\ = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i^2 - 2f_i f_j + f_j^2) \\ = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 \end{aligned}$$

□

2. L is symmetric and positive-semidefinite, since D and W are symmetric and due to 1. $f'Lf \geq 0$ (definition of p.s.d.)

3. The first eigenpair $\lambda_1 = 0$ and $v_0 = \mathbb{1}$ a vector of all ones.

   Proof. $v'Lv = v'\lambda v = \lambda \underbrace{(v'v)}_{\geq 0} \geq 0$, hence $\lambda \geq 0$

   $L\mathbb{1} = [00...0]' = 0\mathbb{1}$ □

4. L has n non-negative real eigenvalues $0 = \lambda_1 \leq \lambda_2 ... \leq \lambda_n$, which follows from 1., 2. and 3.

5. If the graph G has k connected components $\{A_1, ...A_k\}$, then the multiplicity of the eigenvalue 0 is k and the corresponding first eigenvectors are $\{\mathbb{1}_{A_1}, ...\mathbb{1}_{A_k}\}$

   Proof. For the case $k = 1$, i.e. connected graph, if f is an eigenvector with an eigenvalue 0, then

$$0 = f'\lambda f = f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2,$$

   and in order for this to be satisfied f needs to be a constant $f_i = f_j$ in the connected component as all weights $w_{ij} \geq 0$. Without loss of generality let the vertices be ordered succh that connected components are contiguous index intervals, then W and as a consequence L will have a block diagonal form:

$$L = \begin{pmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_k \end{pmatrix},$$

   And each block L_i is a proper Laplacian matrix, corresponding to a connected component. Hence each of those will have exactly one eigenpair with a zero eigenvalue and a corresponding eigenvector $\mathbb{1}_{A_i}$

□

The **Normalized graph Laplacians L_{sym} and L_{rw}** are defined as follows:

$$L_{sym} = D^{-1/2}LD^{-1/2} = I - D^{-1/2}WD^{-1/2}$$

$$L_{rw} = D^{-1}L = I - D^{-1}W$$

They have the following properties:

1. $f \in \mathbb{R}^n, f'L_{sym}f = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2$, which can be shown in a similar way to the first property of L (home assignment for extra credit)

2. (λ, u) is an eigenpair of L_{rw} iff $(\lambda, D^{1/2}u)$ is an eigenpair of L_{sym}

   Proof. By multiply the eigenvalue equation of the symmetric Laplacian $L_{sym}(D^{1/2}u) = \lambda(D^{1/2}u)$ with $D^{-1/2}$ from the left we get

   $$D^{-1/2}L_{sym}(D^{1/2}u) = D^{-1/2}\lambda(D^{1/2}u)$$

   $$D^{-1}LD^{-1/2}D^{1/2}u = \lambda D^{-1/2}D^{1/2}u$$

   $$L_{rw}u = \lambda u$$

   □

3. (λ, u) is an eigenpair of L_{rw} iff (λ, u) solve the generalized eigenvalue problem $Lu = \lambda Du$. To see this we simply multiply the eigenvalue equation $D^{-1}Lu = \lambda u$ with D on the left.

4. 0 is an eigenvalue of L_{rw} with the constant one $\mathbb{1}$ as eigenvector. 0 is an eigenvalue of L_{sym} with a constant eigenvector $D^{1/2}$. First statement obvious, second follows from 2.

5. L_{sym} and L_{rw} are positive semi-definite and have n non-negative real-valued eigenvalues $0 = \lambda_1 \leq ... \leq \lambda_n$. The statement about L_{sym} follows from 1. and for L_{rw} from 2.

6. If the graph G has k connected components $\{A_1, ...A_k\}$, then the multiplicity of the eigenvalue 0 is k for both L_{sym} and L_{rw}. For L_{rw}, the corresponding eigenvectors are $\mathbb{1}_{A_i}$ and those for L_{sym} are $D^{1/2}\mathbb{1}_{A_i}$.

   Proof. Symilar arguments to that for L and using the relations between eigenpairs.

   □

Spectral clustering algorithms

Unnormalized spectral clustering (W, k - number of clusters)

1. Construct L
2. **Compute the first k eigenvectors $\{u1...u_k\}$**
3. Construct U which has $\{u1...u_k\}$ as columns, rows represent vertices
4. Cluster rows of U using k-means into clusters $C_1...C_k$

Normalized spectral clustering using L_{rw} (W, k - number of clusters)

1. Construct L
2. **Compute the first k generalized eigenvectors $\{u1...u_k\}$, where $Lu = \lambda Du$**
3. Construct U which has $\{u1...u_k\}$ as columns, rows represent vertices
4. Cluster rows of U using k-means into clusters $C_1...C_k$

We are computing the generalized eigen vectors and we showed that they are the same as the eigenvalues of L_{rw}.

Normalized spectral clustering using L_{sym} (W, k - number of clusters)

1. Construct L_{sym}
2. **Compute the first k eigenvectors $\{u1...u_k\}$ of L_{sym}**
3. **Construct U which has $\{u1...u_k\}$ as columns, rows represent vertices**
4. **Construct T from U by normalizing the rows to norm 1, i.e.** $t_{ij} = u_{ij}/|u_i|_2$
5. Cluster the rows of T using k-means into clusters $C_1...C_k$

Why does Spectral clustering work?

We want to partition the similarity graph into groups such that edges between group have low weight and those within groups have a high weight. We will show that spectral clustering is an approximation to such graph partitioning problems.

The simplest and most natural graph partitioning is to minimize the cut across partition, where the cut is defined as:

$$cut(A_1...A_k) = \frac{1}{2}\sum_{i=1}^{k} W(A_i, \bar{A}_i),$$

The factor 1/2 is so that we do not count the contribution of every edge twice. This formalization is not very practical as the solution for say $k = 2$ will simply separate one low-degree vertex from the rest oft he graph. Hence, we need a way to balance the clusters. Two common ways: *Rcut* and *Ncut*.

$$Rcut(A_1...A_k) = \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \bar{A}_i)}{|A_i|} = \sum_{i=1}^{k} \frac{cut(A_i, \bar{A}_i)}{|A_i|}$$

$$Ncut(A_1...A_k) = \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \bar{A}_i)}{vol(A_i)} = \sum_{i=1}^{k} \frac{cut(A_i, \bar{A}_i)}{vol(A_i)}$$

These objectives take small values if the clusters are too small. In particular, to minimize $\sum 1/|A_i|$ we want $|A_i| = |A_j|$, and similarly for $\sum 1/vol(A_i)$ we want equality of volumes.

Minimizing Rcut and Ncut is NP-hard and spectral clustering is a way to solve relaxed versions of these problems. We will show that relaxing Ncut leads to normalized spectral clustering, and Rcut to unnormalized spectral clustering.

Theorem 4. ***[Rayleigh-Ritz]*** *Let M be a symmetric matrix, then*

$$\lambda_k = min_{x \perp v_1,..,v_{k-1}} \frac{x'Mx}{x'x} = max_{x \perp v_{k+1},..,v_n} \frac{x'Mx}{x'x}$$

and the extreme value are attained at $x = v_k$. In particular, if x maximizes the Rayleigh quotient

$$x = argmax_x \frac{x'Mx}{x'x},$$

where $x'x \neq 0$, then $(\frac{x'Mx}{x'x}, x)$ is the "largest" eigenpair.

Proof. We can assume that x is a unit vector w.l.g., because if $y = ax, a \in \mathbb{R}$, then $\frac{y'My}{y'y} = \frac{ax'Max}{ax'ax} = \frac{x'Mx}{x'x}$. So $x'x = 1$. Eigenvalues of M are indexed from small to large $\lambda_1 \leq \lambda_2... \leq \lambda_n$ and the corresponding eigenvectors are $\{v_1...v_n\}$. From the spectral theorem $\{v_i\}$ forms an orthonormal basis in $\mathbb{R}^n$, so any vector, including x can be expressed as:

$$x = \sum_i (v_i'x)v_i.$$

[Mini-proof 1:] To see the above we multiply the RHS by another eigenvector v_j' from the left to get:

$$v_j' \sum_i (v_i'x)v_i = \underbrace{\sum_i (v_i'x)v_j'v_i}_{linearity} = \underbrace{(v_j'x)v_j'v_j}_{i \neq j \iff v_i'v_j = 0} = \underbrace{v_j'x}_{v_j'v_j = 1}$$

Also, due to the above representation of x we can also show the following:

$$1 = x'x = (\sum_i (v_i'x)v_i)'(\sum_j (v_j'x)v_j) = \sum_{i,j} (v_i'x)(v_j'x)v_i'v_j = \sum_j (v_j'x)^2 \tag{4}$$

$$\begin{aligned}
\frac{x'Mx}{x'x} &= x'Mx & , x'x = 1 \\
&= (\sum_i (v_i'x)v_i)'M(\sum_j (v_j'x)v_j) & , x = \sum_i (v_i'x)v_i \\
&= (\sum_i (v_i'x)v_i)'(\sum_j (v_j'x)\lambda_j v_j) & , Mv_j = \lambda_j v_j \\
&= \sum_{i,j} (v_i'x)(v_j'x)\lambda_j v_i'v_j & , distr.sum \\
&= \sum_j (v_j'x)^2 \lambda_j & , i \neq j \iff v_i'v_j = 0
\end{aligned}$$

Using (1) and the derivation above, we get the following extremes for the Rayleigh quotient, showing the extreme cases $k = 1$ and $k = n$:

$$\frac{x'Mx}{x'x} = \sum_j (v_j'x)^2 \lambda_j \geq \sum_j (v_j'x)^2 \lambda_1 = \lambda_1, \tag{5}$$

and

$$\frac{x'Mx}{x'x} = \sum_j (v_j'x)^2 \lambda_j \leq \sum_j (v_j'x)^2 \lambda_n = \lambda_n, \tag{6}$$

Since the eigenvectors are orthogonal, we can get the general case for arbitrary k.

□

Next we will reason about the relationship between spectral clustering and minimizing Rcut and Ncut.

Approximating Rcut, $k = 2$

Our goal is to solve the optimization problem:

$$\min_{A \subset V} Rcut(A, \bar{A})$$

We will show that we can re-write this objective in terms of the graph Laplacian.

Given a set $A \subset V$, define the vector f as:

$$f_i = \begin{cases} \sqrt{|\bar{A}|/|A|} & \text{if } v_i \in A \\ -\sqrt{|A|/|\bar{A}|} & \text{if } v_i \in \bar{A} \end{cases} .$$

Now the Rcut objective can be re-written as:

$$\begin{aligned}
f'Lf &= \frac{1}{2}\sum_{i,j} w_{ij}(f_i - f_j)^2 && \text{, Prop 1. for } L \\
&= \frac{1}{2}\sum_{i\in A, j\in\bar{A}} w_{ij}\Big(\sqrt{\frac{|\bar{A}|}{|A|}} + \sqrt{\frac{|A|}{|\bar{A}|}}\Big)^2 + \frac{1}{2}\sum_{i\in\bar{A}, j\in A} w_{ij}\Big(-\sqrt{\frac{|\bar{A}|}{|A|}} - \sqrt{\frac{|A|}{|\bar{A}|}}\Big)^2 && \text{, Def. of } f \\
&= cut(A,\bar{A})\Big(\frac{|\bar{A}|}{|A|} + \frac{|A|}{|\bar{A}|} + 2\Big) \\
&= cut(A,\bar{A})\Big(\frac{|\bar{A}|+|A|}{|A|} + \frac{|\bar{A}|+|A|}{|\bar{A}|}\Big) \\
&= (|A|+|\bar{A}|)Rcut(A,\bar{A}) \\
&= |V|Rcut(A,\bar{A})
\end{aligned}$$

Additionally, we can show that f is orthogonal to $\mathbb{1}$:

$$f'\mathbb{1} = \sum_i f_i = \sum_{i\in A}\sqrt{\frac{|\bar{A}|}{|A|}} - \sum_{i\in\bar{A}}\sqrt{\frac{|A|}{|\bar{A}|}} = |A|\sqrt{\frac{|\bar{A}|}{|A|}} - |\bar{A}|\sqrt{\frac{|A|}{|\bar{A}|}} = 0.$$

And

$$f'f = |f|_2^2 = \sum_i f_i^2 = |A|\frac{|\bar{A}|}{|A|} + |\bar{A}|\frac{|A|}{|\bar{A}|} = |A| + |\bar{A}| = n = |V|$$

Hence the problem of minimizing Rcut is equivalent to:

Definition 2. ***[Rcut minimization in terms of** L**]***

$$\min_{A\subset V} f'Lf,\ \ s.t.\ \ f\perp\mathbb{1}, f_i \textit{ defined as above}, |f|_2 = \sqrt{n}$$

This is discrete optimization problem since values of f can take two distinct values and is still NP-hard. The simplest relaxation is to discard the discrete condition, i.e. allow f to take arbitrary values obtaining:

Definition 3. ***[Relaxed Rcut minimization in terms of** L**]***

$$\min_f f'Lf,\ \ s.t.\ \ f\perp\mathbb{1}, |f|_2 = \sqrt{n} \iff \min_{f\perp\mathbb{1}}\frac{f'Lf}{f'f}$$

Based on the Relay-Ritz theorem, the solution to the relaxed version is the second "smallest" eigenvector. So we can approximate a minimizer of Rcut by v_2 of L. In order to get a solution for Rcut we need to transform the obtained f into a two-value indicator for the partitions. Once can use the sign, but more generally once can simply cluster the values. When using more than one partitions we will cluster the corresponding vectors. This is exactly the unnormalized Spectral clustering algorithm.
Rcut, $k > 2$
To show a general k equivalence one follows similar development. Now, we have multiple partitions $A_1..A_k$ and there will be k indicator vectors $h_j = (h_{1j}...h_{nj})$:

$$h_{ij} = \begin{cases} 1/\sqrt{|A_j|} & \text{if } v_i \in A_j \\ 0 & \text{if } v_i \in \bar{A}_j \end{cases},$$

where $i = 1..n$ and $j = 1..k$. We set up the matrix $H_{n\times k}$ containing the indicator vectors as columns. The columns are orthonormal, i.e. $H'H = I$,and similar to the calculations above we can show that:

$$h_i'Lh_i = cut(A_i,\bar{A}_i)/|A_i|$$

,
and moreover $h_i'Lh_i = (H'LH)_{ii}$, and combining those facts:

$$Rcut(A_1...A_k) = Tr(H'LH),$$

where the Tr is the trace of a matrix. Hence, minimizing Rcut is equivalent to

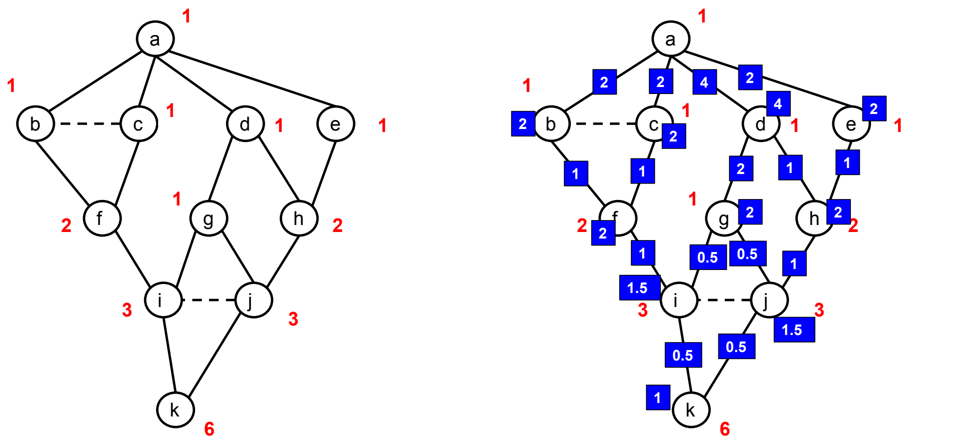
$$min_H Tr(H'LH), s.t. H'H = I, H \text{ is defined as above}$$

Dropping the discrete values for H results in a relaxation that asks for the first k eigenvectors of L (Note: H is U in the algorithm.)

5.3 Girvan-Newman BC computation

The GN algorithms is an $O(|V||E|)$ algorithm to compute the betweeness centrality of every edge. It proceeds as follows:

1. Input: graph $G(V, E)$
2. Output: the number of pairs whose SP pass through every edge $c_{x,y}, \forall (x, y) \in E$
3. Repeat for every vertex $v \in V$
   (a) Compute BFS DAG, disregard edges on the same level
   (b) $s_v^v = 1$
   (c) Top-Down: compute #SP from v to every nodes $s_x^v = \sum_{y \in parents(x)} s_y^v$ (1.a)
   (d) Bottom-Up: compute credit for nodes $c_x^y = \begin{cases} 1 & \text{if } x \text{ is leaf} \\ 1 + \sum_{z \in children(x)} c_{x\to z}^y & \text{if } v_i \in \bar{A}_j \end{cases}$ and edges: $c_{x\to z}^y = c_z^y \frac{s_x^y}{\sum_{j \in parents(z)} s_j^y}$ (1.b)
4. $c_{x,y} = \sum_{v \in V} (c_{x\to y}^v + c_{y\to x}^v)$



(a) Top-Down: Compute SPs $s_x^a, \forall x$ (b) Bottom-Up: Compute credit: c_x^a and $c_{x\leftarrow z}^a$

Figure 3: Girvan-Newman's BC computation.

5.4 Modularity and configuration models

Modularity is another measure of "goodness" of graph partitioning. It takes into account the expected number of edges among members in the community in a random realization of the observed network. The main question one needs to answer is: what is a random realization?

Configuration model

Given an unweighted undirected graph $G(V, E)$, consider all random graphs: $G'(V, E')$ with the same nodes and preserved node degrees $d_v = |N(v)|$. To obtain those graphs, one can imagine the following random process:

1. cut every edge resulting in 2 “stub” edges adjacent to the corresponding nodes
2. re-connect stubs to form full edges randomly, thus preserving d_v

Properties:

1. The expected number of edges between nodes u and v in G' is $E[\#Edges(u, v)] = d_u \frac{d_v}{2m}$, normalization is the total number of stubs
2. The total number of expected edges in G' is

$$E[|E'|] = \frac{1}{2} \sum_{i,j \in V} \frac{d_i d_j}{2m} = \frac{1}{4m} \sum_i d_i \underbrace{\sum_j d_j}_{2m} = \frac{2m2m}{4m} = m$$

Let C be a partitioning of G, then the modiularity is defined as:

$$Q(C) = \frac{1}{2m} \sum_{c \in C} \sum_{i,j \in c} (x_{ij} - \frac{d_i d_j}{2m}),$$

where $x_{ij} = 1 \iff (i, j) \in E$. Modularity is between $[-1, 1]$. It is positive when the edges in communities exceed the expected number of edges.

References

[1] Philippe Flajolet and G Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences*, 31(2):182–209, 1985.