1. **Use hierarchical clustering to get clusters**

The observation matrices of project-skill and student-skill is input to the clustering algorithms to generate the hierarchy of clusters. Here the observation matrix is binary matrix with project/students as rows and skills as columns, where 1 in cell [I,j] represents ith student/project possess/need the jth skill and 0 in cell [I,j] represents ith student/projects doesn’t possess/need the jth skill. The hierarchical clustering algorithm will generate the hierarchy of clusters from these matrices. For example we have student-skill, projects-skill matrix for k-skills.

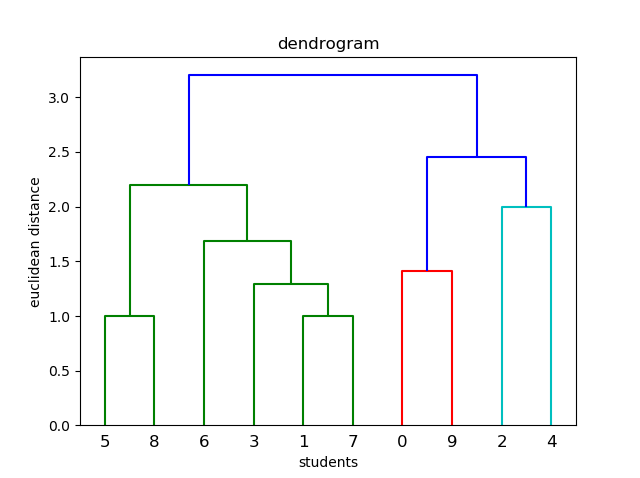
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | data integration | frequent patterns | supervised learning | neural chips | core memory | manufacturing systems | flexible structures | probability | m-p model | internet |
| Student0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| Student1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Student2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| Student3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Student4 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| Student5 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| Student6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Student7 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Student8 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Student9 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

Student-skill matrix (m students – k skills)

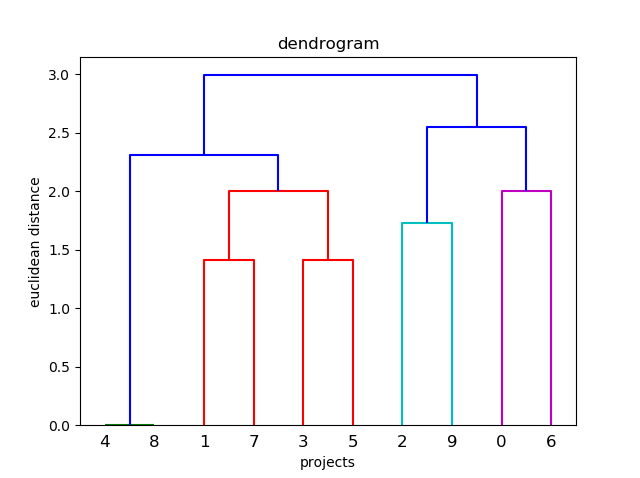
Project-skill matrix ( n projects – k skills )

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | data integration | frequent patterns | supervised learning | neural chips | core memory | manufacturing systems | flexible structures | probability | m-p model | internet |
| project 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| project 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| project 2 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| project 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| project 4 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| Project 5 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Project 6 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| Project 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Project 8 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| project 9 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |

The hierarchy created for student-skill matrix



The hierarchy created for project-skill matrix



The hierarchical clustering tool will generate the clusters for both matrices based on the data.

For example, the tool will generate 2 clusters for student-skill matrix and 2 clusters for projects-skill matrix based on the Euclidean distance. Then the clusters will be converted in to binary matrix with clusters as row and projects/students and skills as column.

Clusters for project-skill

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Sk0 | Sk1 | Sk2 | Sk3 | Sk4 | Sk5 | Sk6 | Sk7 | Sk8 | Sk9 | P0 | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 |
| C0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| C1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |

Clusters for student-skill

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Sk0 | Sk1 | Sk2 | Sk3 | Sk4 | Sk5 | Sk6 | Sk7 | Sk8 | Sk9 | S0 | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 |
| C0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| C1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |

Where p0, p1… are project 0, project 1 …

S0, s1… are student 0, student 1 …

Sk0, sk1… are skill 0, skill 1 (data integration, frequent patterns, …)

C0, c1 are clusters

1. **Generate affinity between each row of generated concept matrices**

Now, we will generate the affinity between each row of student\_skill\_cluster matrix with each row of project\_skill\_ matrix according to the skill that are present in both concepts.

Let we are calculating affinity between ith student\_skill\_cluster and jth project\_skill\_cluster then,

1. If some skill Sk is present in both cluster. This shows that Sk skill is required by all project in jth project\_skill\_cluster and is present in all students of the ith student\_skill\_cluster.
2. If some skill Sk is not present in either of cluster. This shows that Sk is nor required in the projects nor possesses by the students.
3. If some skill Sk is present in ith student\_skill\_cluster but not in jth project\_skill\_cluster.This is a case where the skill is present in all students of ith cluster but it is not required by any of the project in jth cluster.
4. If some skill Sk is present in jth project\_skill\_cluster but not in ith student\_skill\_cluster. This is the case where the skills is not present in students of ith cluster but it is required by all the projects in jth clusters.

Case a gives the most favourable match since it satisfies a core skill requirement of one or more tasks. The second case is also favourable since it avoids an extra skill which would remain underutilized. The last two cases are unfavourable matches. The mutual affinities between projects and students are calculated according to the above guidelines and stored in the aff[p][q]. below is the pseudo-code for generating the affinity matrix. Constants a, b, c and d are user input positive constants empirically determined to suit the nature of the contexts.

**For each** cluster i in Student\_skill\_Cluster **do** {

**For each** cluster j in Project\_skill\_Cluster **do** {

Affi[i][j]:= 0

**For each** skill Sk in Skill{1..k} **do** {

**If** (student\_skill\_cluster[i][k] = 0

**AND** project\_skill\_cluster[j][k] = 1) **then**

Aff[i][j] := Aff[i][j] + a\*student\_skill\_cluster[i][k]

**Else if** (student\_skill\_cluster[i][k] = 0

**AND** project\_skill\_cluster[j][k] = 0) **then**

Aff[i][j] := Aff[i][j] + b

**Else if** (student\_skill\_cluster[i][k] = 0

**AND** project\_skill\_cluster[j][k] = 0) **then**

Aff[i][j] := Aff[i][j] - c\*student\_skill\_cluster[i][k]

**Else**

Aff[i][j] := Aff[i][j] - d }

}

}

}

this will give us the affinity matrix Aff[p][q] where (I,j) position shows how much ith student cluster prefer the jth project cluster or how much jth project cluster prefers the ith student cluster.

Example:

Aff matrix :

|  |  |  |
| --- | --- | --- |
|  | Project c0 | Project c1 |
| Student c0 | 2 | 6 |
| Student c1 | 4 | -4 |

Now we can generate the preferences for each student cluster and project cluster as the values in the ith row suggests how much ith student cluster prefers the jth project cluster and values in the jth row suggests how much jth project cluster prefers the ith student cluster.

Preferences for student cluster and project cluster are:

student preferences

stu c0 [proj c1, proj c0]

stu c1 [proj c0, proj c1]

project preferences

proj c0 [stu c1, stu c0]

proj c1 [stu c0, stu c1]

1. **Apply extended stable marriage algorithm**

Now, we have preference order we can apply project cluster oriented extended stable marriage algorithm. The algorithm will give us the best matching pair of concepts. The project cluster oriented extended stable marriage algorithm is given below:

**While** P in project\_clusters and P’s list contain S not allotted to P **do** {

s := first S not allotted to H

**if** s is Assigned to some other project\_cluster P’ **then**

break assignment (s, P’)

assign s to P

**for each** successor P’ of P in s’s list **do** {

remove P’ and s from each others list

}

}

For our example this will give us list pairs that is :

[ (0, 1), (1, 0) ]

Where for every pair (j,i) j refer to jth project cluster and I refer to ith student cluster.

Stability for each pairs are (based on common skills) :

for pair ( 0 , 1 )

stable percentage: 0.625

for pair ( 1 , 0 )

stable percentage: 0.75

---------------------------------------------------------------------------

average stable percentage: 0.6875

---------------------------------------------------------------------------

Now, every student in ith student cluster can contribute to tasks of every jth project in project cluster that require skills set present in that cluster.

List of students that can contribute to each projects are:

Format > Project ( total students included, % of skills covered ) > list of students

p0 ( 6 , 0.5 ) >{s1,s3,s5,s6,s7,s8,}

p1 ( 4 , 1.0 ) >{s0,s9,s2,s4,}

p2 ( 6 , 1.0 ) >{s1,s3,s5,s6,s7,s8,}

p3 ( 4 , 1.0 ) >{s0,s9,s2,s4,}

p4 ( 4 , 1.0 ) >{s0,s9,s2,s4,}

p5 ( 4 , 1.0 ) >{s0,s9,s2,s4,}

p6 ( 6 , 0.2 ) >{s1,s3,s5,s6,s7,s8,}

p7 ( 4 , 1.0 ) >{s0,s9,s2,s4,}

p8 ( 4 , 1.0 ) >{s0,s9,s2,s4,}

p9 ( 6 , 0.8 ) >{s1,s3,s5,s6,s7,s8,}

List of projects in which each students can contribute to are:

Format > student ( total projects the student get included, % of skills utilized ) > list of projects

s0 ( 6 , 0.8 ) >{p1,p3,p4,p5,p7,p8,}

s1 ( 6 , 1.0 ) >{p0,p9,p2,p6,}

s2 ( 6 , 1.0 ) >{p1,p3,p4,p5,p7,p8,}

s3 ( 6 , 1.0 ) >{p0,p9,p2,p6,}

s4 ( 6 , 0.8 ) >{p1,p3,p4,p5,p7,p8,}

s5 ( 6 , 1.0 ) >{p0,p9,p2,p6,}

s6 ( 6 , 1.0 ) >{p0,p9,p2,p6,}

s7 ( 6 , 1.0 ) >{p0,p9,p2,p6,}

s8 ( 6 , 1.0 ) >{p0,p9,p2,p6,}

s9 ( 6 , 0.7 ) >{p1,p3,p4,p5,p7,p8,}

1. **Limiting the involvement of students in lots of projects**

For each student, Calculate the significance for each project allocated to the student as per follow:

1. If student’s skills is not required in the project, assign significance[project] := -1 (least signif.)
2. If student’s skills requirement in the projects can be fulfil by other students included in the project, then assign

Significance[project] := total skills - no. of common skills between the student and other student in the project (no of common skill will decrease signif.)

1. if student possess unique skills in group that are required in project, then assign highest significance

significance[project] := total skills + no. of unique skills

remove the projects allotted to the student starting from least significant project to max. till project is less than or equal to limit\_val and project doesn’t requires student’s unique skills.

After limiting,

List of students that can contribute to each projects are:

Format > Project ( total students included, % of skills covered ) > list of students

p0 ( 2 , 0.5 ) >{s7,s8,}

p1 ( 2 , 1.0 ) >{s9,s4,}

p2 ( 3 , 1.0 ) >{s1,s3,s5,}

p3 ( 1 , 1.0 ) >{s2,}

p4 ( 1 , 1.0 ) >{s4,}

p5 ( 1 , 1.0 ) >{s4,}

p6 ( 2 , 0.2 ) >{s1,s6,}

p7 ( 1 , 1.0 ) >{s0,}

p8 ( 3 , 1.0 ) >{s0,s9,s2,}

p9 ( 5 , 0.8 ) >{s3,s5,s6,s7,s8,}

List of projects in which each students can contribute to are:

Format > student ( total projects the student get included, % of skills utilized ) > list of projects

s0 ( 2 , 0.6 ) >{p7,p8,}

s1 ( 2 , 1.0 ) >{p2,p6,}

s2 ( 2 , 0.7 ) >{p3,p8,}

s3 ( 2 , 1.0 ) >{p9,p2,}

s4 ( 3 , 0.8 ) >{p1,p4,p5,}

s5 ( 2 , 1.0 ) >{p9,p2,}

s6 ( 2 , 1.0 ) >{p9,p6,}

s7 ( 2 , 1.0 ) >{p0,p9,}

s8 ( 2 , 1.0 ) >{p0,p9,}

s9 ( 2 , 0.7 ) >{p1,p8,}