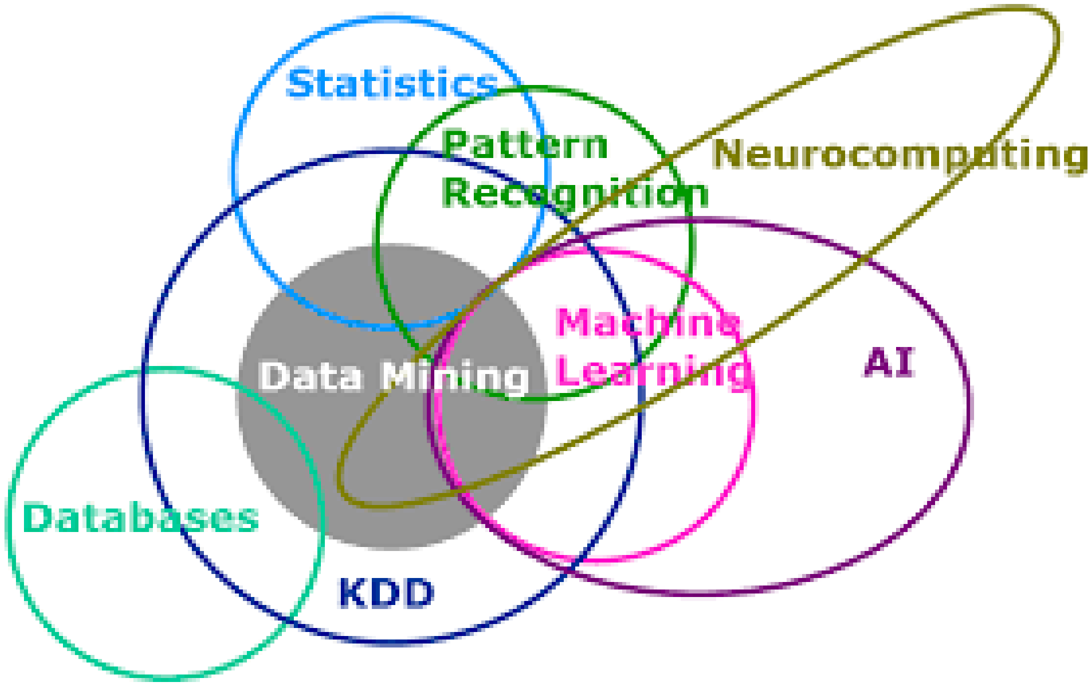
In the name of Allah the most Beneficial ever merciful

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ



Artificial Intelligence (AI) in Software Engineering

Feature Space concept

K-means

Department of Computer Science , Univeristy of Karachi (DCS-UBIT)

8th June 2021

Motivation

Feature Selection
Feature Normalization
Feature Representation

1

JM1 is listed as highly imbalanced datasets in experimental study

SN Computer Science (2020) 1:108
https://doi.org/10.1007/s42979-020-0119-4

Check for updates

ORIGINAL RESEARCH

SN

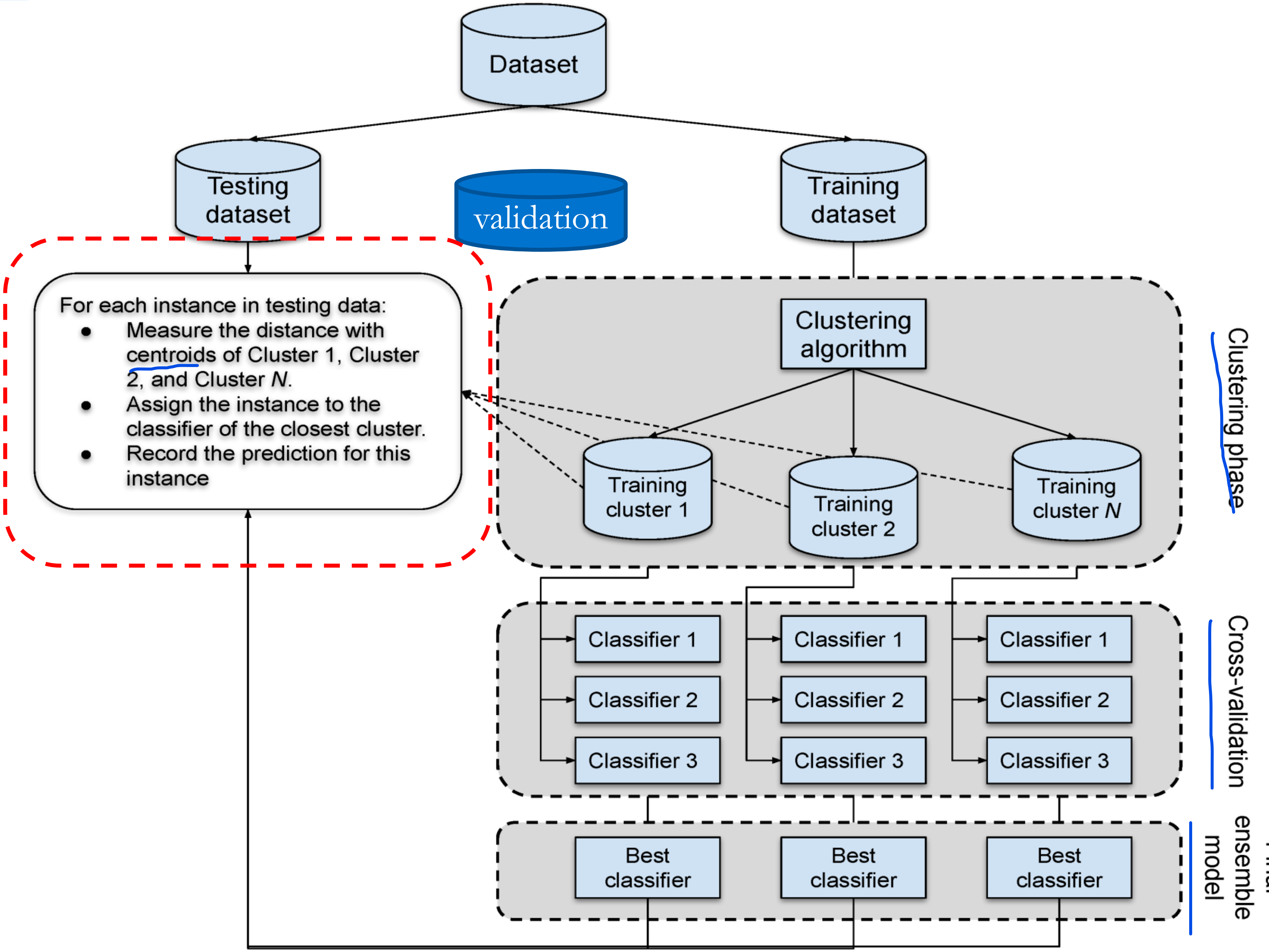
Evaluation of Sampling-Based Ensembles of Classifiers on Imbalanced Data for Software Defect Prediction Problems

Thanh Tung Khuat[1] · My Hanh Le[2]

Received: 21 September 2019 / Accepted: 11 March 2020

UNIVERSITY OF KARACHI

Proposed Methodology : Ensemble Learning based Classifier



Representing Data for Classification

- ✓ we shall assume that data provided to us as feature vectors.

- ✓ Each data point is represented as a d-dimensional vector (d numbers).

- ✓ we assume data points are provided as $\overline{\boldsymbol{x_1}}, \overline{\boldsymbol{x_2}}, \overline{\boldsymbol{x_3}} \ldots\ldots\ldots \overline{\boldsymbol{x_n}}$

  where each $\boldsymbol{x_t} \in \mathbb{R}^{\boldsymbol{d}}$ is a d-dimensional vector

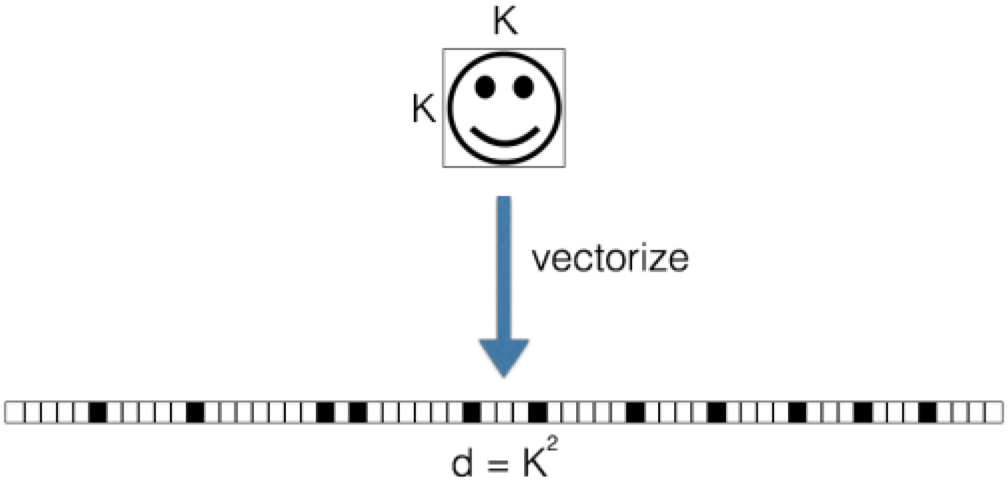
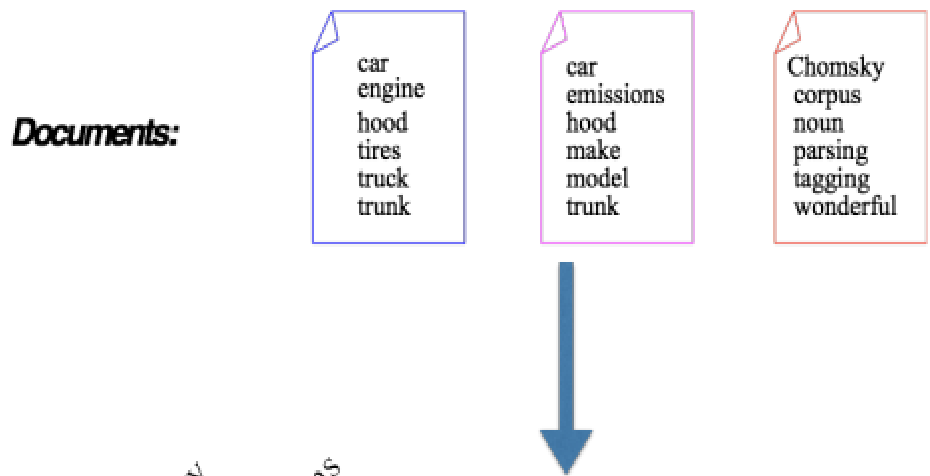


Figure Image as vector

- ✓ Each document is represented by a vector
- ✓ Dimensionality is that of number of words in the lexicon (number of words in the English dictionary for English documents for example).

171,146 words



car	Chomsky	corpus	emissions	engine	hood	make	model	noun	parsing	tagging	tires	truck	trunk	wonderful
1	0	0	0	1	1	0	0	0	0	0	1	1	1	0
1	0	0	1	0	1	1	1	0	0	0	0	0	1	0
0	1	1	0	0	0	0	0	1	1	1	0	0	0	1

Figure 2: Bag of words feature for documents

- ✓ Each entry of the vector for a given document, corresponds to one word in the lexicon
- ✓ The value for that entry is the number of times the word appears in the given document.

UNIVERSITY OF KARACHI

Example of Feature vector from software Engineering perspective

Deep Semantic Feature Learning for Software Defect Prediction

Song Wang, Taiyue Liu, Jaechang Nam, and Lin Tan

Abstract—Software defect prediction, which predicts defective code regions, can assist developers in finding bugs and prioritizing their testing efforts. Traditional defect prediction features often fail to capture the semantic differences between different programs. This degrades the performance of the prediction models built on these traditional features. Thus, the capability to capture the semantics in programs is required to build accurate prediction models. To bridge the gap between semantics and defect prediction features, we propose leveraging a powerful representation-learning algorithm, deep learning, to learn the semantic representations of programs automatically from source code files and code changes. Specifically, we leverage a deep belief network (DBN) to automatically learn semantic features using token vectors extracted from the programs' abstract syntax trees (AST) (for file-level defect prediction models) and source code changes (for change-level defect prediction models).

We examine the effectiveness of our approach on two file-level defect prediction tasks (i.e., file-level within-project defect prediction and file-level cross-project defect prediction) and two change-level defect prediction tasks (i.e., change-level within-project defect prediction and change-level cross-project defect prediction). Our experimental results indicate that the DBN-based semantic features can significantly improve the examined defect prediction tasks. Specifically, the improvements of semantic features against existing traditional features (in F1) range from 2.1 to 41.9 percentage points for file-level within-project defect prediction, from 1.5 to 13.4 percentage points for file-level cross-project defect prediction, from 1.0 to 8.6 percentage points for change-level within-project defect prediction, and from 0.6 to 9.9 percentage points for change-level cross-project defect prediction.

Index Terms—Defect prediction, quality assurance, deep learning, semantic features.

[tse-2877612-pp.pdf (purdue.edu)](tse-2877612-pp.pdf)

UNIVERSITY OF KARACHI

Features: Complexity, function call, programming token

For example, Figure 1 shows an original buggy version, i.e., Figure 1(a), and a fixed clean version, i.e., Figure 1(b), of a method from Lucene. In the buggy version, there is an IOException when initializing variables `os` and `is` before the try block. The buggy version can lead to a memory leak[1] and has already been fixed by moving the initializing statements into the try block in Figure 1(b). Using traditional features to represent these two code snippets, e.g., code complexity features, their feature vectors are identical. This is because these two code snippets have the same source code characteristics in terms of complexity, function calls, raw programming tokens, etc. However, the semantic information in these two code snippets is significantly different. Specifically, the contextual information of the two variables, i.e., `os` and `is`, in the two versions is different.

```
public void copy(Directory to, String src, String dest)
      throws IOException {
IndexOutput os = to.createOutput(dest);
IndexInput is = openInput(src);
IOException priorException = null;

try {
    is.copyBytes(os, is.length());
} catch (IOException ioe) {
     priorException = ioe;
}
finally {
     IOUtils.closeSafely(priorException, os, is);
  }
}
```

(a) Original buggy code snippet.

UNIVERSITY OF KARACHI

Proposed semantic feature vectors

```java
public void copy(Directory to, String src, String dest)
      throws IOException {
IndexOutput os = to.createOutput(dest);
IndexInput is = openInput(src);
IOException priorException = null;

try {
    is.copyBytes(os, is.length());
} catch (IOException ioe) {
     priorException = ioe;
}
finally {
     IOUtils.closeSafely(priorException, os, is);
  }
}
```

(a) Original buggy code snippet.

```java
public void copy(Directory to, String src, String dest)
      throws IOException {
IndexOutput os = null;
IndexInput is = null;
IOException priorException = null;
try {
   os = to.createOutput(dest);
   is = openInput(src);
   is.copyBytes(os, is.length());
} catch (IOException ioe) {
     priorException = ioe;
} finally {
     IOUtils.closeSafely(priorException, os, is);
  }
}
```

(b) Code snippet after fixing the bug.

Fig. 1: A motivating example from Lucene.

WPDP and CPDP

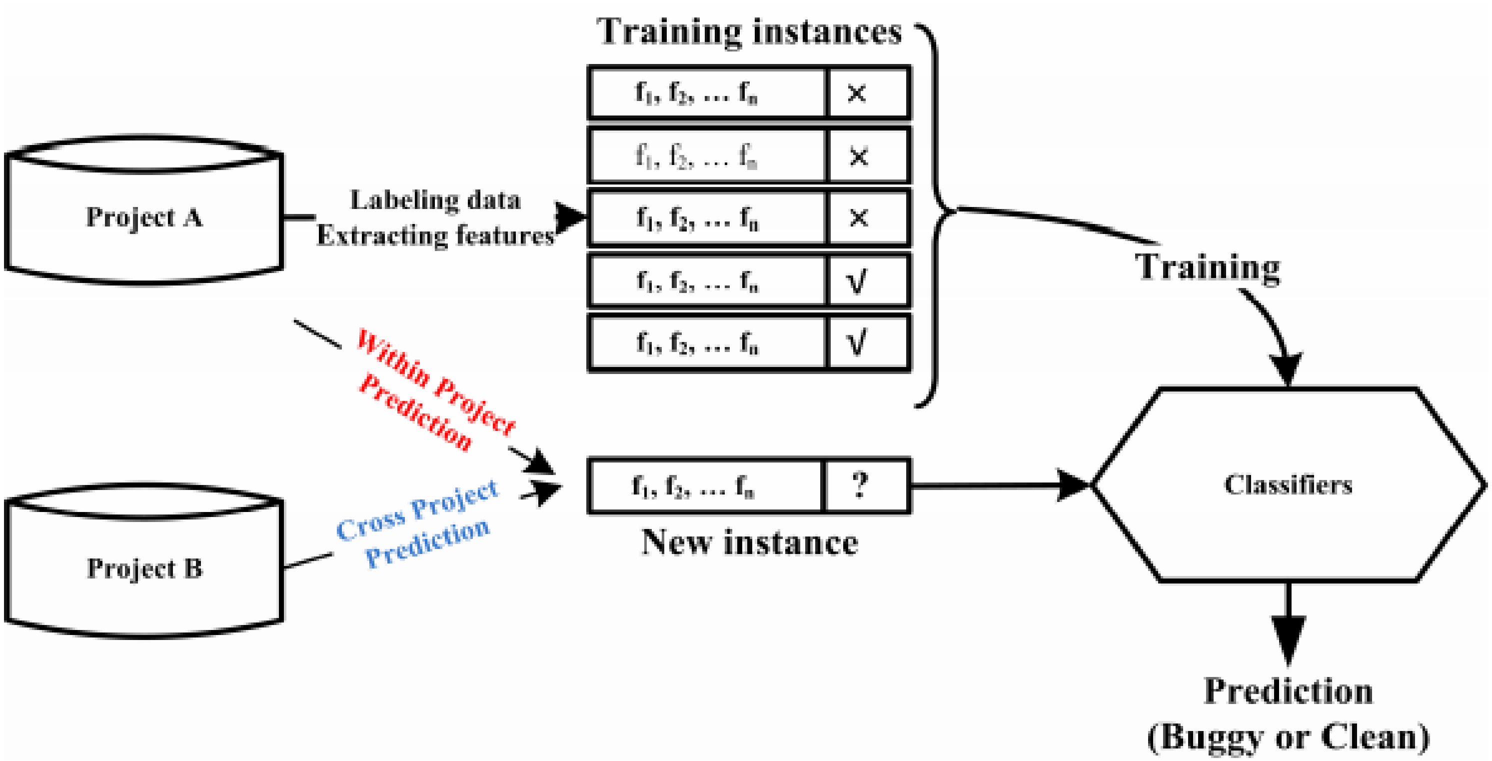


Fig. 2: Defect Prediction Process

Explain the difference between buggy and clean feature vector

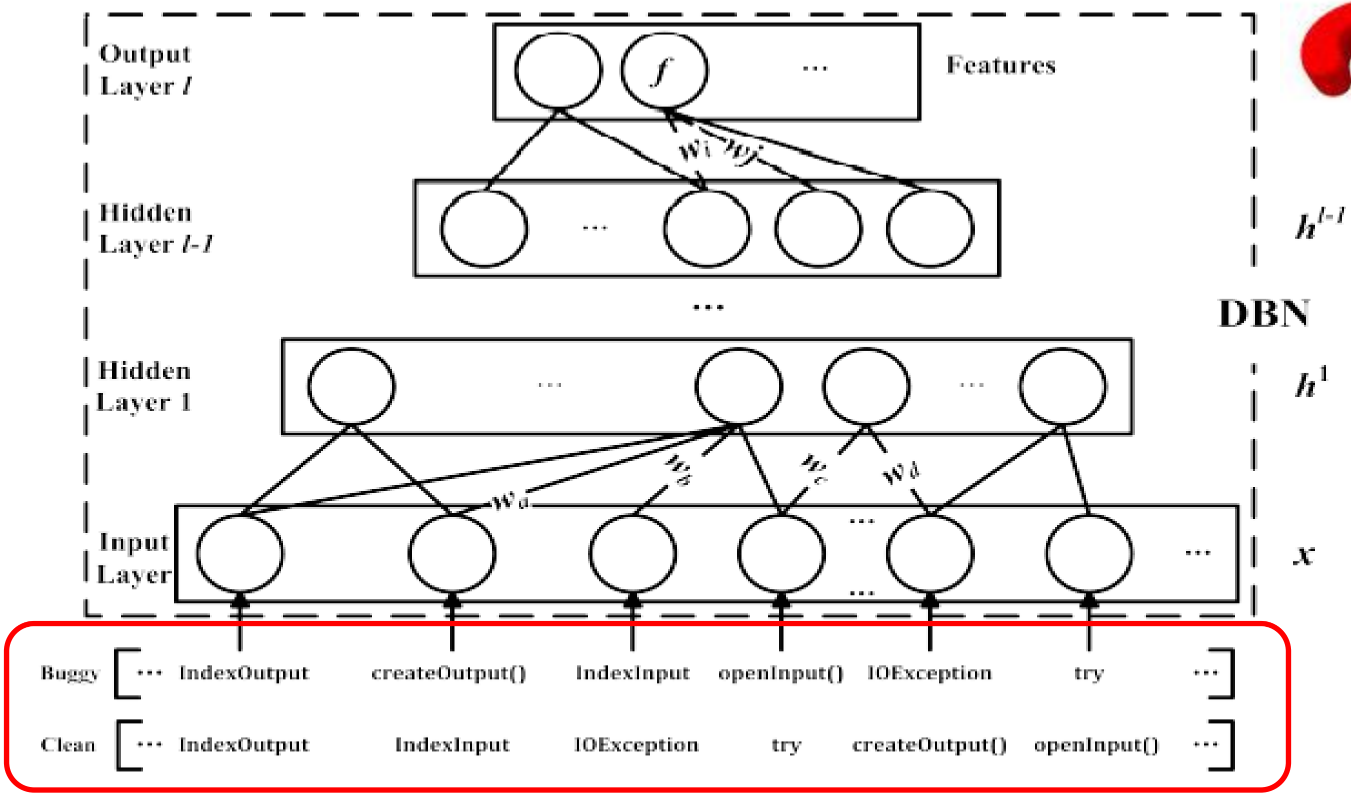


Fig. 3: Deep belief network architecture and input instances of the buggy version and the clean version presented in Figure 1. Although the token sets of these two code snippets are identical, the different structural and contextual information between tokens enables DBN to generate different features to distinguish them.

UNIVERSITY OF KARACHI

DBN-based approach to generating semantic features

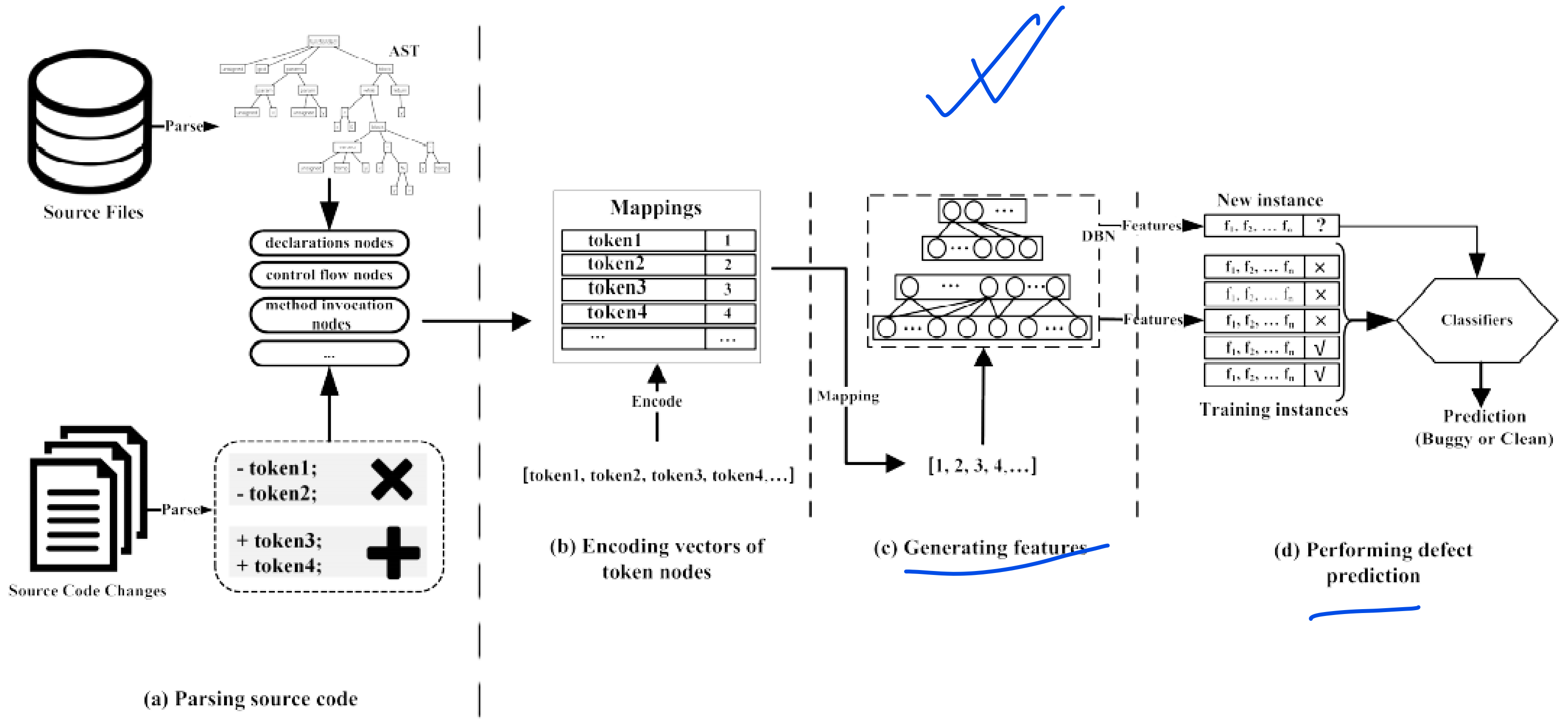


Fig. 5: Overview of our DBN-based approach to generating semantic features for file-level and change-level defect prediction.

UNIVERSITY OF KARACHI

Evaluation Metrics

Precision and recall are composed of three numbers in terms of *true positive*, *false positive*, and *false negative*. True positive is the number of predicted defective files (or changes) that are truly defective, while false positive is the number of predicted defective ones that are actually not defective. A false negative records the number of predicted non-defective files (or changes) that are actually defective. Higher precision is demanded by developers who do not want to waste their debugging efforts on the non-defective code, while higher recall is often required for mission-critical systems, e.g., revealing additional defects [112]. However, comparing defect prediction models by using only these two metrics may be incomplete. For example, one could simply predict all instances as buggy instances to achieve a recall score of 1.0 (which will likely result in a low precision score) or only classify the instances with higher confidence values as buggy instances to achieve a higher precision score (which could result in a low recall score). To overcome the above issues, we also use the F1 score (i.e., F1), which is the harmonic mean of precision and recall.

UNIVERSITY OF KARACHI

Precision, Recall and F1

$$Precision = \frac{true\ positive}{true\ positive + false\ positive} \qquad (6)$$

$$Recall = \frac{true\ positive}{true\ positive + false\ negative} \qquad (7)$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \qquad (8)$$

K-Means

(UnSupervised Classifier)

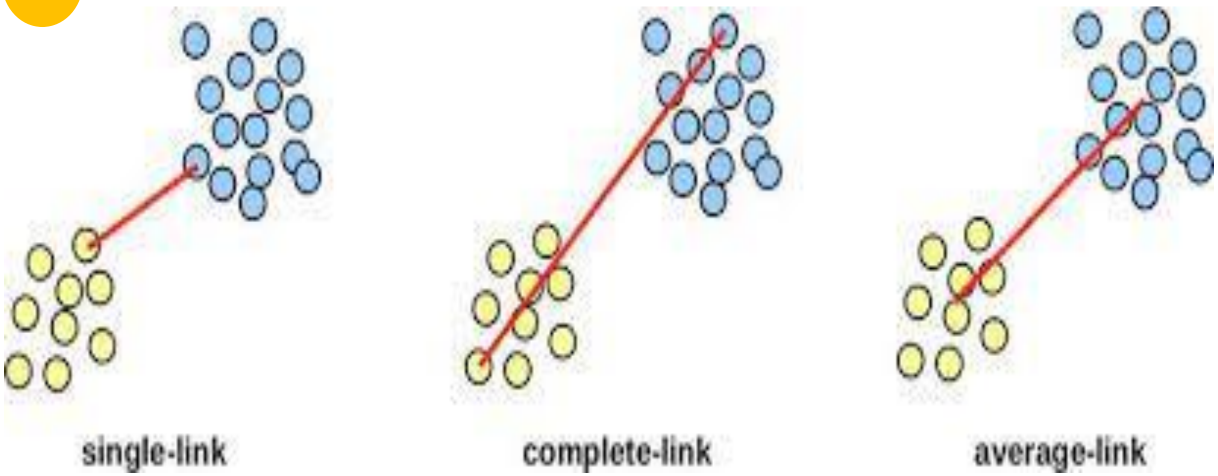
Un-Supervised Learning

- ✓ Centroid Models (K-Means)
- ✓ Connectivity Models (Hierarchical clustering)
- ✓ Density Models (DBSCAN)

  Density-Based Spatial **Clustering** of Applications with Noise
- ✓ Graph based Models
- ✓ Sub-Space based Models
- ✓ Feature Extraction (PCA, ICA , SVD)

UNIVERSITY OF KARACHI

Single-link and complete-link clustering



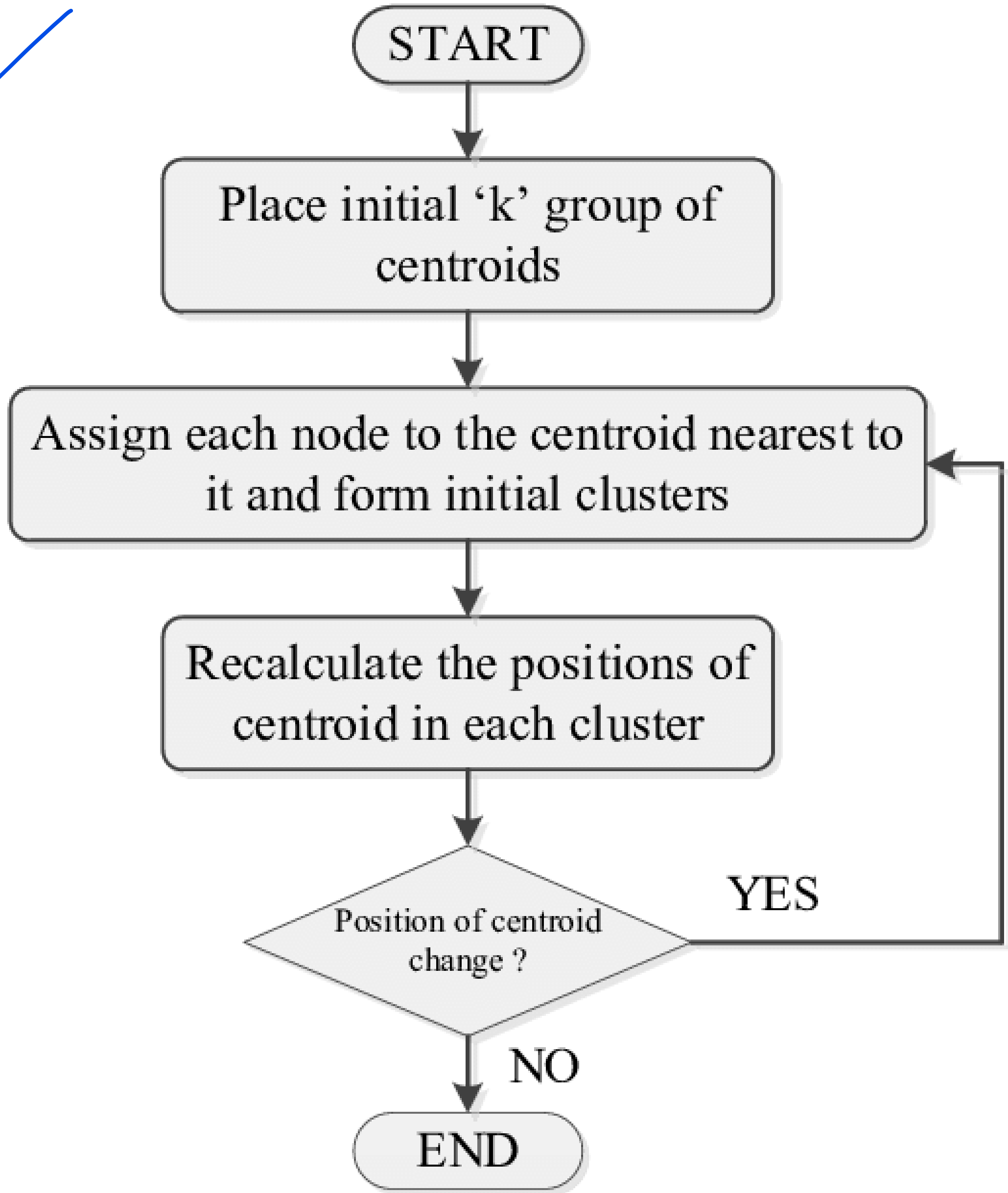
Which metric is used to judge similarity or dissimilarity ??

- ✓ The similarity of two clusters is the similarity of their *most similar* members

- ✓ The similarity of two clusters is the similarity of their *most dissimilar* members

UNIVERSITY OF KARACHI

K-Means Clustering Algorithm Flow Chart

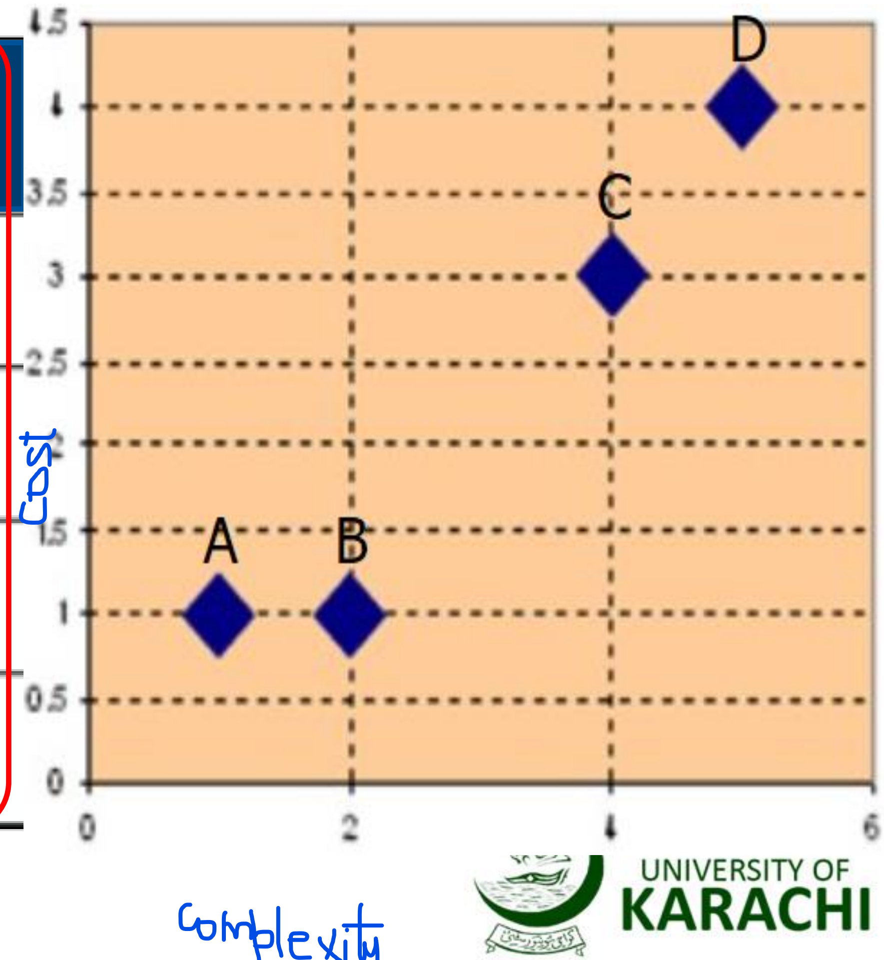


UNIVERSITY OF KARACHI

Step I → Data Matrix, Visualization

- ✓ Suppose we have 4 types of projects, and each has two attributes (complexity, cost)

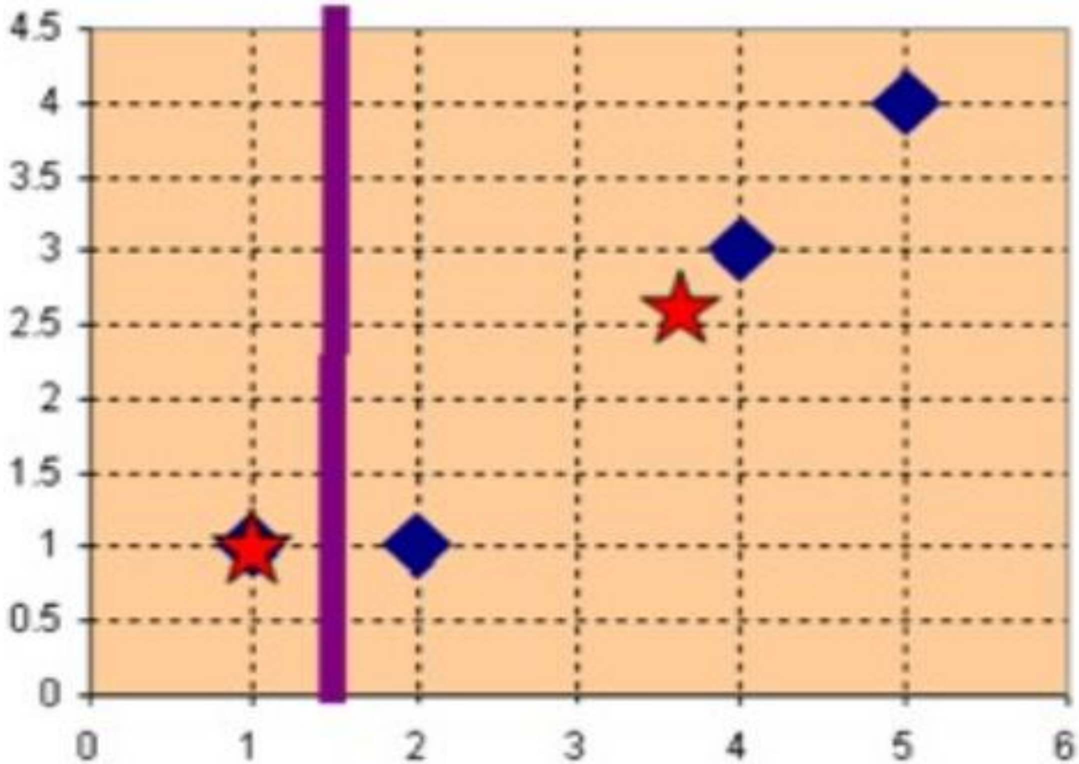
Project	Complexity (Level)	Cost (Millions)
A	1	1
B	2	1
C	4	3
D	5	4



Step II : Visualization / Intuition

Where are centroids ??

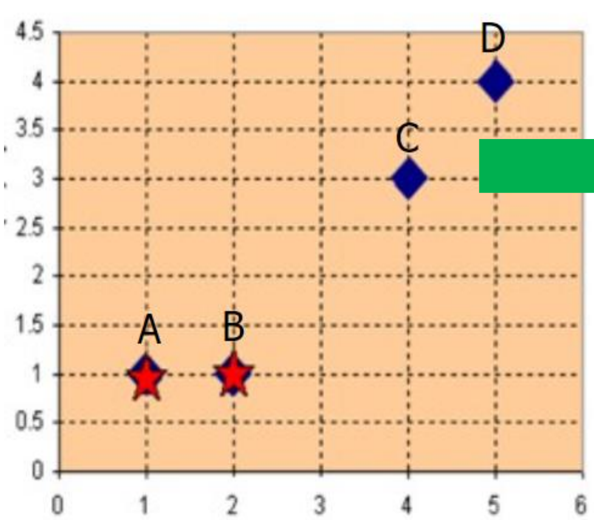
Are you satisfied with partitioning between data points ??



Step III: Use Initial Seed Points for partitioning

Project	complexity	Cost (Millions)
A	1	1
B	2	1
C	4	3
D	5	4

iteration 0



$$c_1 = A,\ c_2 = B$$

$$\mathbf{D}^0 = \begin{bmatrix} 0 & 1 & 3.61 & \boxed{5} \\ 1 & 0 & 2.83 & \boxed{4.24} \end{bmatrix} \qquad \mathbf{c}_1 = (1,1) \quad group-1$$

$$\mathbf{c}_2 = (2,1) \quad group-2$$

Euclidean distance

$$\begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \end{array} \begin{array}{c} \\ X \\ Y \end{array}$$

$$d(D,c_1) = \sqrt{(5-1)^2 + (4-1)^2} = 5$$

$$d(D,c_2) = \sqrt{(5-2)^2 + (4-1)^2} = 4.24$$

Assign each object to the cluster with the nearest seed point

UNIVERSITY OF KARACHI

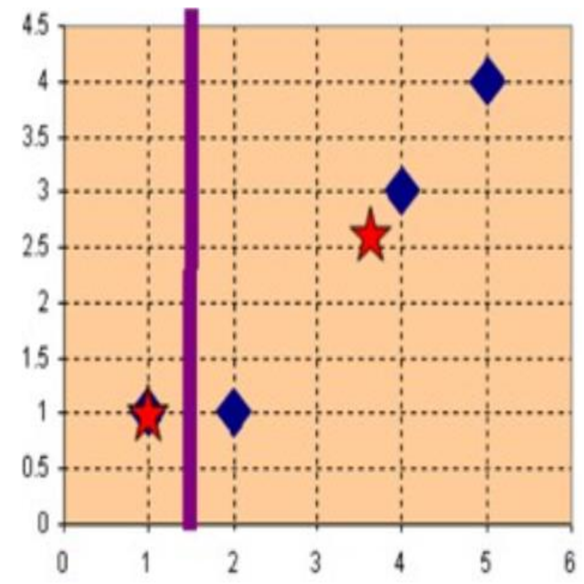
Step IV: Update centroid

$$c_1 = (1,\ 1)$$

$$c_2 = \left(\frac{2+4+5}{3}, \frac{1+3+4}{3} \right)$$

$$= (11/3,\ 8/3)$$

$$= (3.67,\ 2.67)$$



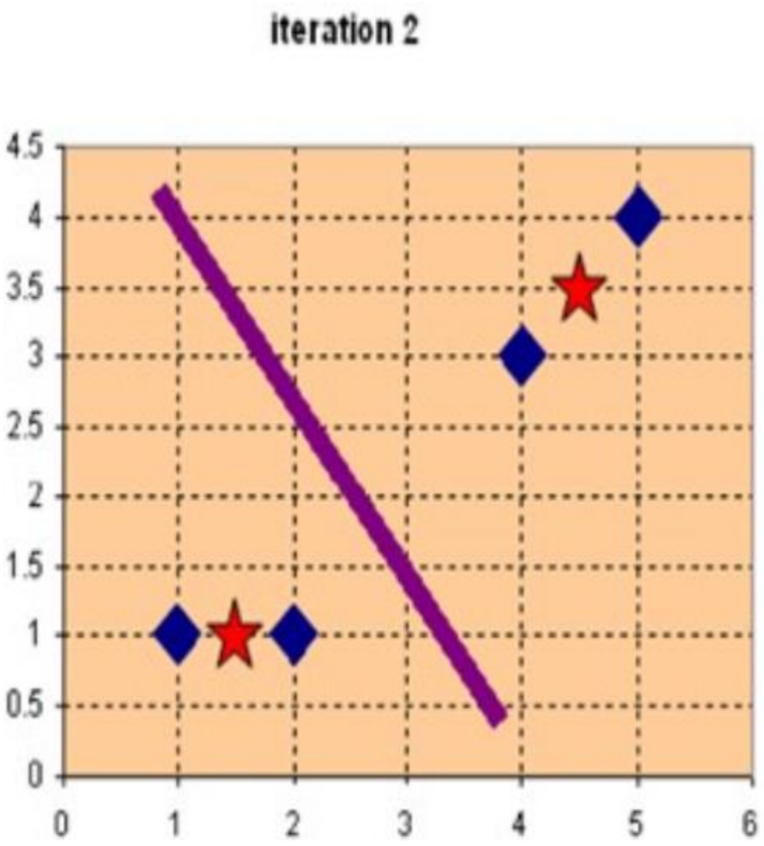
UNIVERSITY OF KARACHI

Step V: Should we continue or Stop

Knowing the members of each cluster, now we compute the new centroid of each group based on these new memberships.

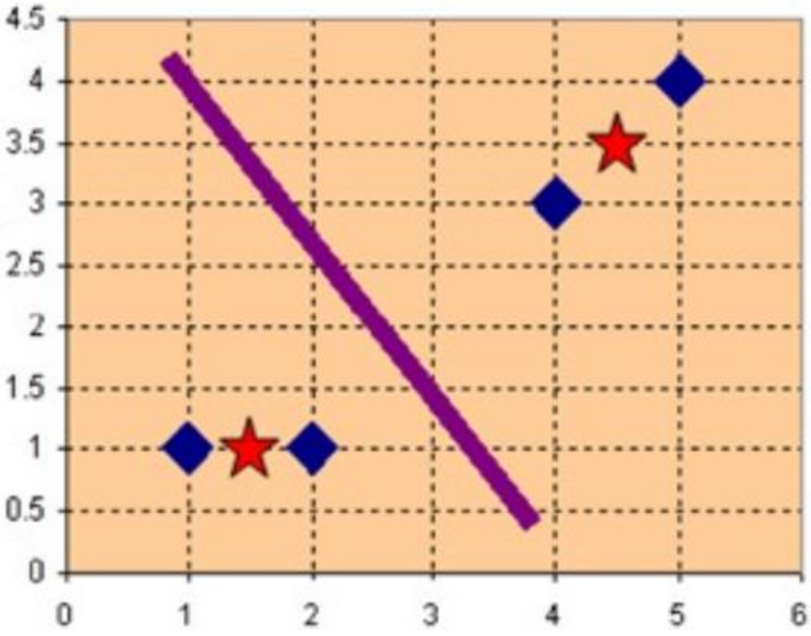
$$c_1 = \left(\frac{1+2}{2}, \frac{1+1}{2}\right) \Rightarrow \left(1\frac{1}{2},\ 1\right)$$

$$c_2 = \left(\frac{4+5}{2}, \frac{3+4}{2}\right) \Rightarrow \left(4\frac{1}{2},\ 3\frac{1}{2}\right)$$



UNIVERSITY OF KARACHI

Step VI : Stop due to convergence



Compute the distance of all objects to the new centroids

$$\mathbf{D}^2 = \begin{bmatrix} 0.5 & 0.5 & 3.20 & 4.61 \\ 4.30 & 3.54 & 0.71 & 0.71 \end{bmatrix} \quad \begin{array}{ll} \mathbf{c}_1 = (1\frac{1}{2}, 1) & \mathit{group}-1 \\ \mathbf{c}_2 = (4\frac{1}{2}, 3\frac{1}{2}) & \mathit{group}-2 \end{array}$$

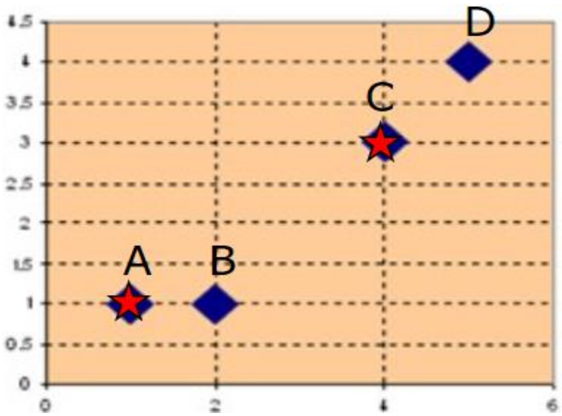
$$\begin{array}{cccc} A & B & C & D \end{array}$$

$$\begin{array}{c} \\ \\ \end{array}\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \begin{array}{l} X \\ Y \end{array}$$

Stop due to no new assignment

Question to Answer

- ✓ How many steps are needed for convergence ??
- ✓ What are members of two clusters ??
- ✓ What are centroids of two cluster after convergence ??



UNIVERSITY OF
KARACHI

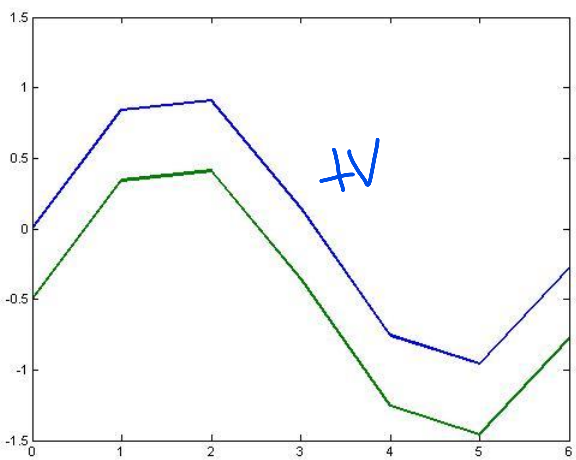
Pearson Linear Correlation
(Feature vector as Profile)

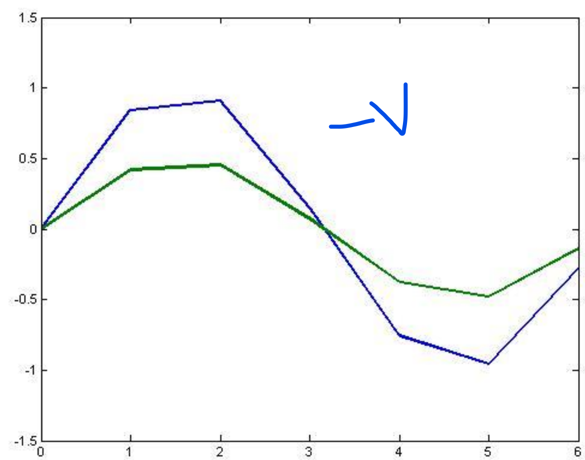
PLC in feature space

Which metric is used to judge similarity or dissimilarity ??

- ✓ We might care more about the overall shape of expression profiles rather than the actual magnitudes

- ✓ That is, we might want to consider similar when they are "up" and "down" together





UNIVERSITY OF KARACHI

Weekly Assignment

Submission with declaration and signature is compulsory,

Prepare two Examples of dummy feature vectors

- ✓ Choose 1 example code snippet as we discussed in lecture
- ✓ Prepare their clean and buggy version
- ✓ Prepare Feature vector

☐ I don’t understand a single word about Feature vector

☐ If not, What is the main hindrance to start work?

☑ If Yes, share your experience/practice/work

UNIVERSITY OF KARACHI

Practice Confusion Matrix, Precision, Recall , F1

- ✓ Find a toy example on Precision, Recall, F1.
- ✓ Solve/Redo it in your own handwriting
- ✓ Tick following Check box and submit

- [ ] I don't understand a single word about Precision, Recall, F1.
- [ ] I can solve toy problems like this
- [ ] I understand and explain/present it to my friends.

Practice K-Means Toy Exampe

- ✓ Find relevant Example????
- ✓ Must show all working steps using data matrix and distance matrix
- ✓ Submit work with declaration

☐ I don’t understand a single word about K-Means

☐ If not, What is the main hindrance to start work?

☐ If Yes, share your experience/practice/wrok

UNIVERSITY OF KARACHI

Can we some how apply K-Means on J1 Dataset for meaning-ful grouping/ classification

✓ Find relevant Code/Dataset/ ????

✓ Submit code/Analysis with strong feedback

☐ I don’t understand a single word about K-Means

☐ If not, What is the main hindrance to start work?

☐ If Yes, share your experience/practice/work

UNIVERSITY OF KARACHI