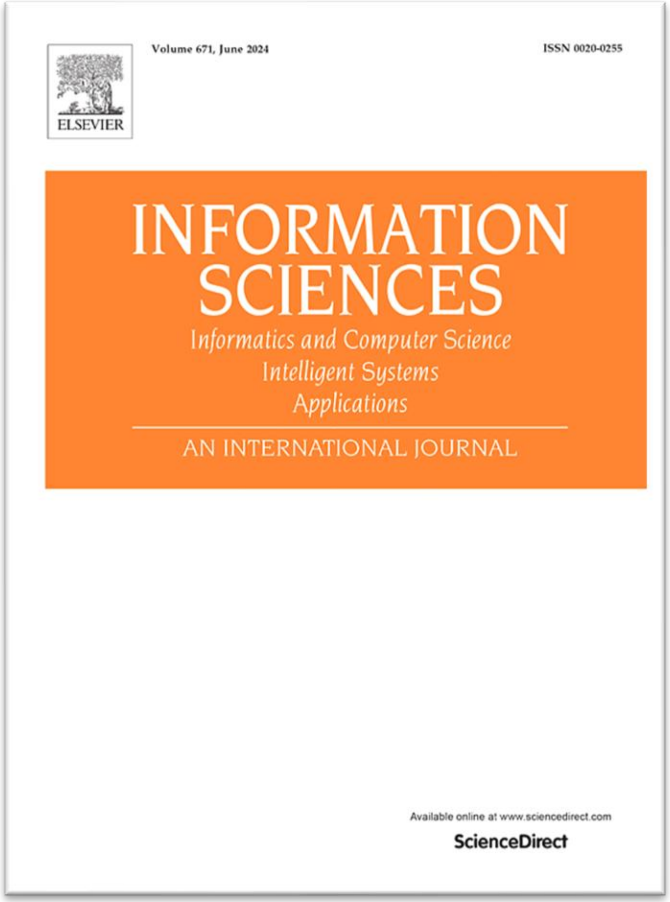
Ensemble *k*-nearest neighbors based on centroid displacement



Author: Alex X. Wang, Stefanka S. Chukova,
Binh P. Nguyen (New Zealand)

Journal: Information Science (Elsevier)
Volume 629, 2023
https://www.sciencedirect.com/science/article/pii/S0020025523001731

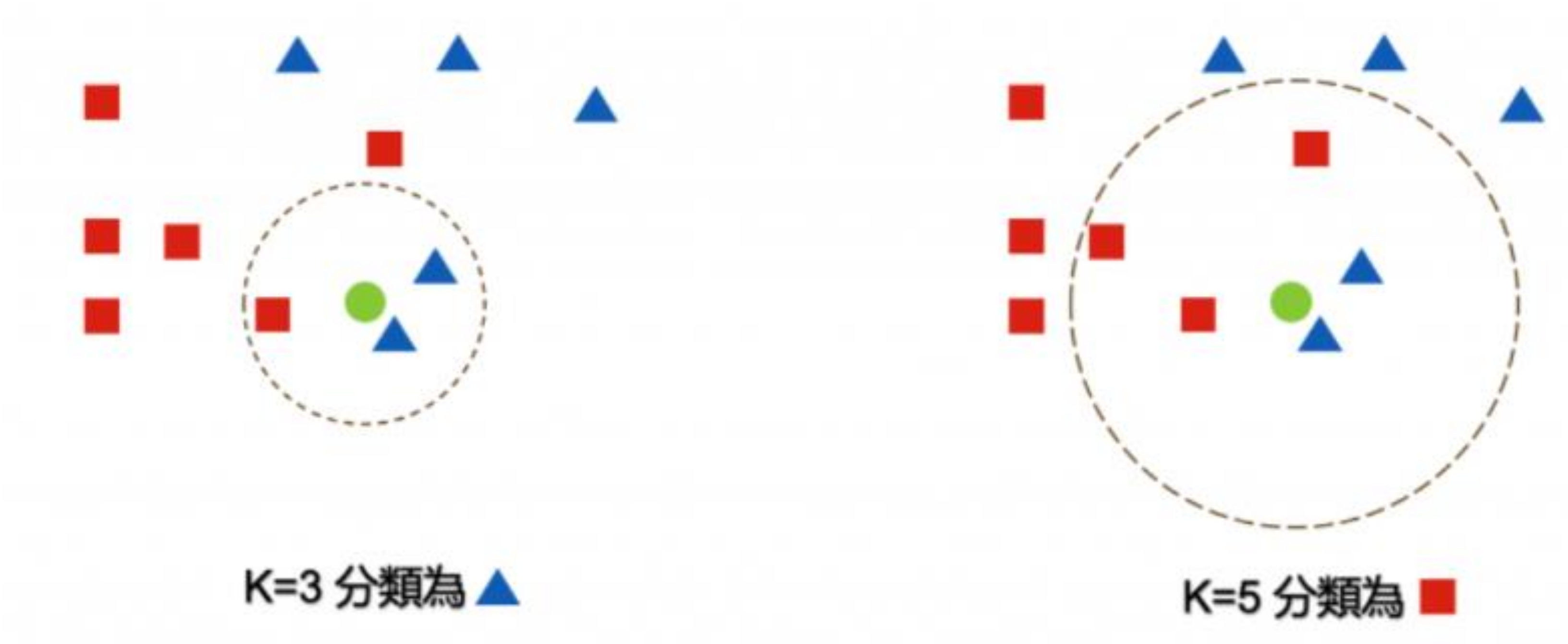
Prof. Wei-Mei Chen
Student: Chang-En Lyu (M11202117)

Agenda

- Introduction
- Definition
- Algorithm
- Experiment
- My Experiment
- Conclusion

Introduction

k-NN classification



https://ithelp.ithome.com.tw/m/articles/10269826

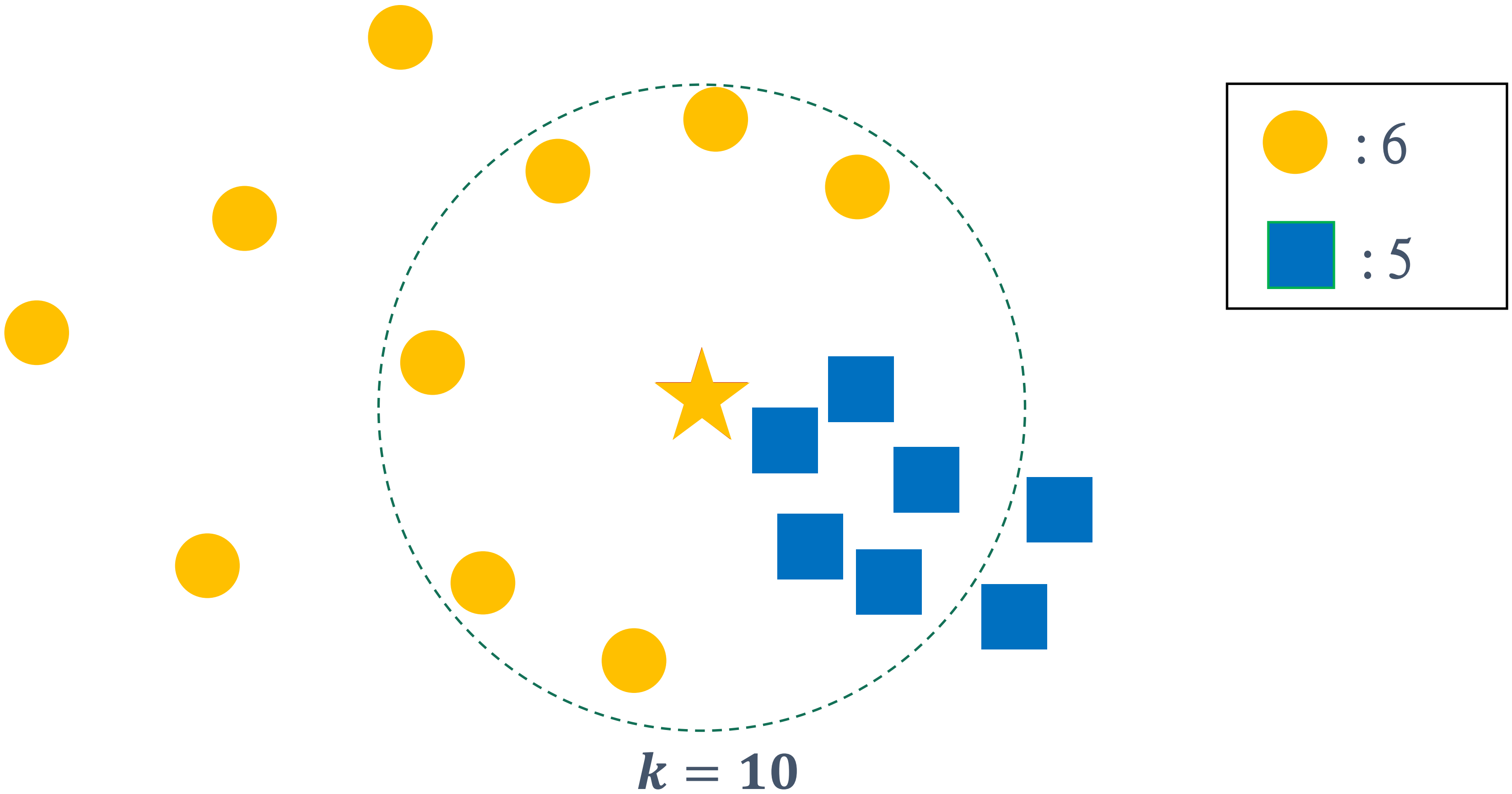
Advantage

- Conceptual Simplicity
- Strong Generalization Performance on Different Datasets

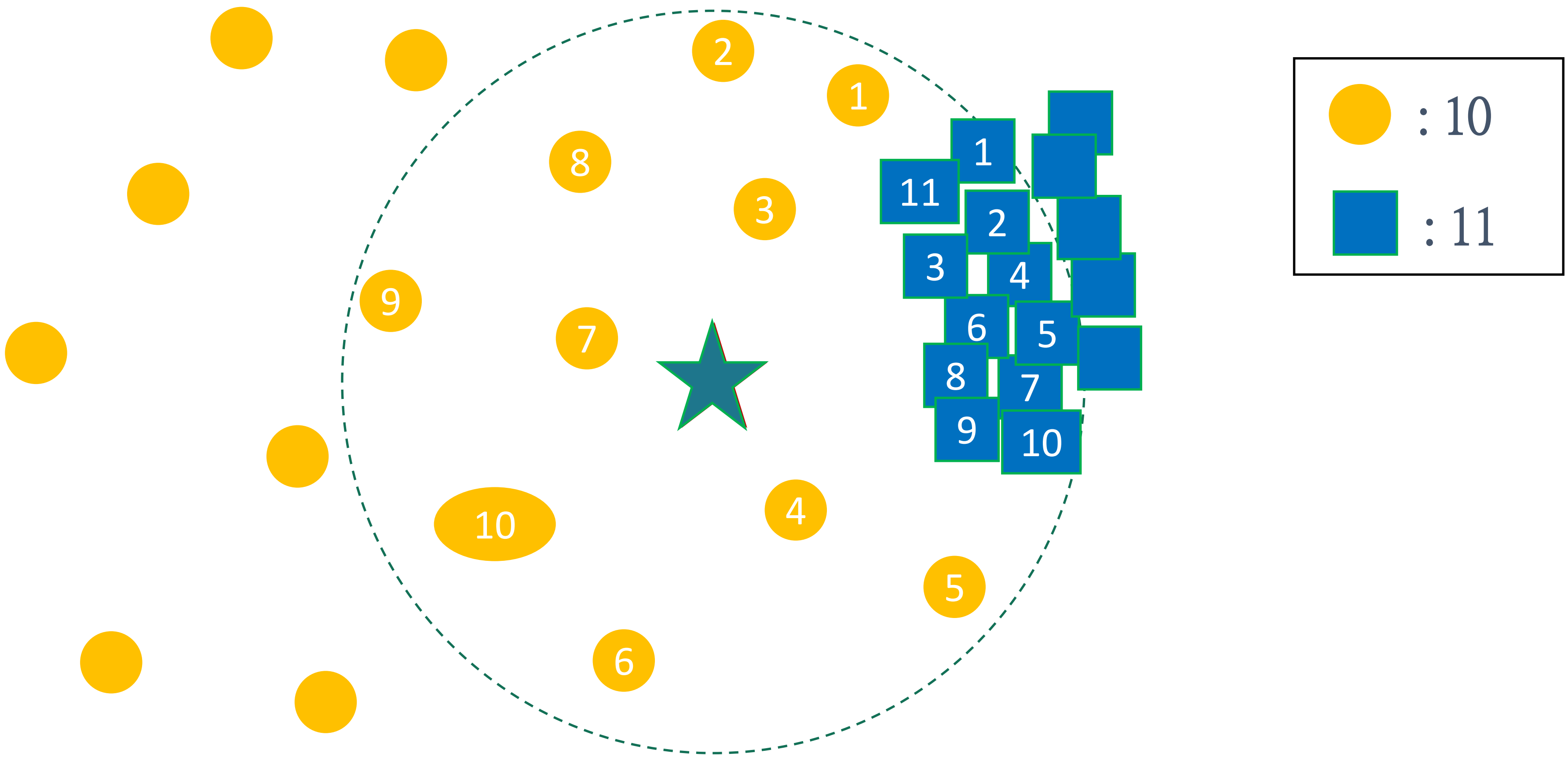
Disadvantage

- Arbitrarily selecting neighborhood size k
- Computation Challenge of High-Dimension Data
- **Simply Majority Voting Rule**

k-NN Simply Majority Voting Rule



k-NN Simply Majority Voting Rule

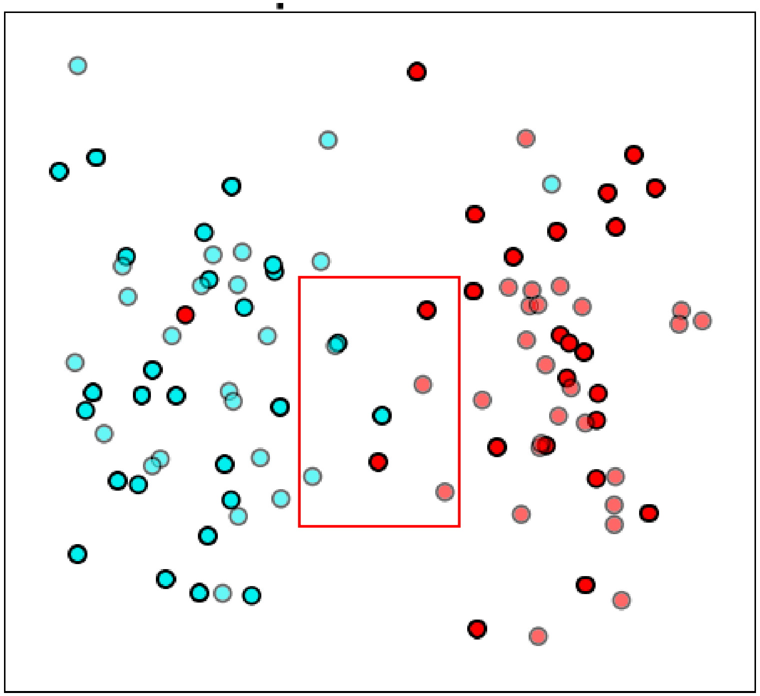


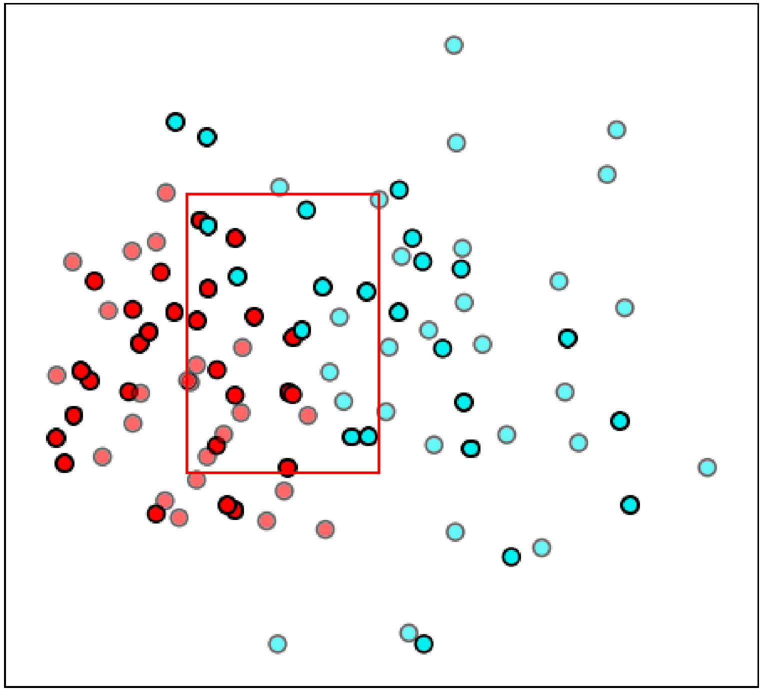
$k = 21$

$\boldsymbol{k}$-NN Simply Majority Voting Rule

- Where does the problem occur
  - Varying Densities (heterogeneous)
  - More Complexity dataset
  - Boundary

Occurring Scene





Improvement Method

- CDNN
  - Solve the problem above
  - Inspired by the k-means algorithm
  - More time-consuming than original KNN
- **ECDNN (Proposed Algorithm)**
  - Less time-consuming than CDNN
  - It considers the **confidence** between KNNs

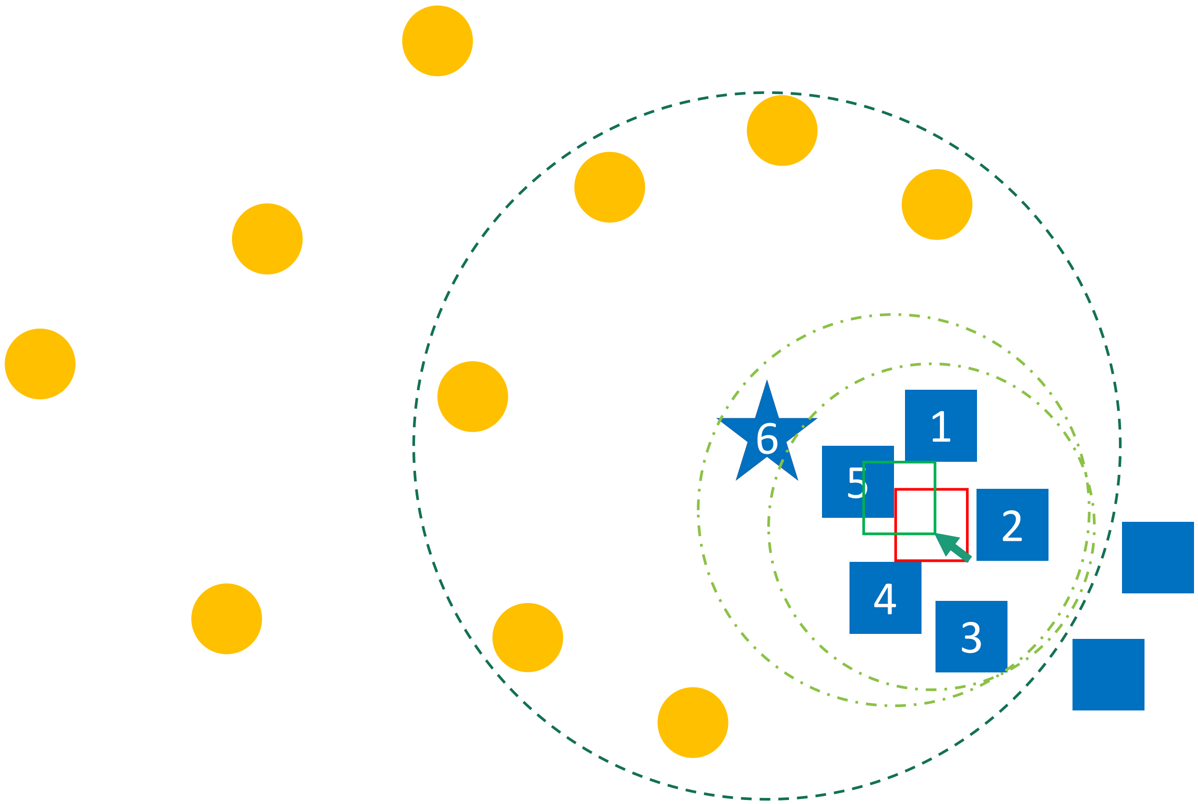
Definition

Centroid Displacement

- Centroid
  - Each cluster exists a **representative point**. a common choice being the **mean** (also called the **centroid**) of all points in the cluster,
    $$\mu_i = \frac{1}{n_i} \sum_{x_j \in C_i} x_j$$
  - <u>We need to find the **minimum displacement** within its k-NN clusters.</u>

Centroid Displacement

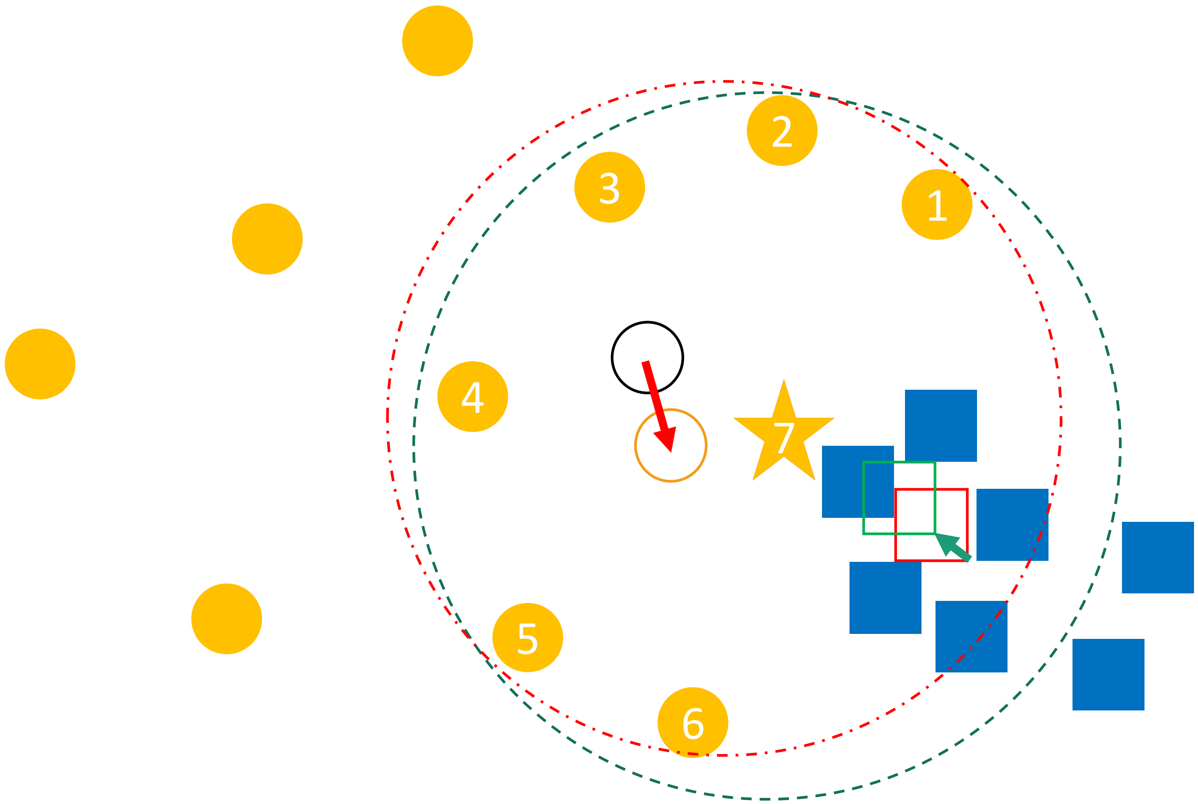
Blue Class



$k = 10$

Centroid Displacement

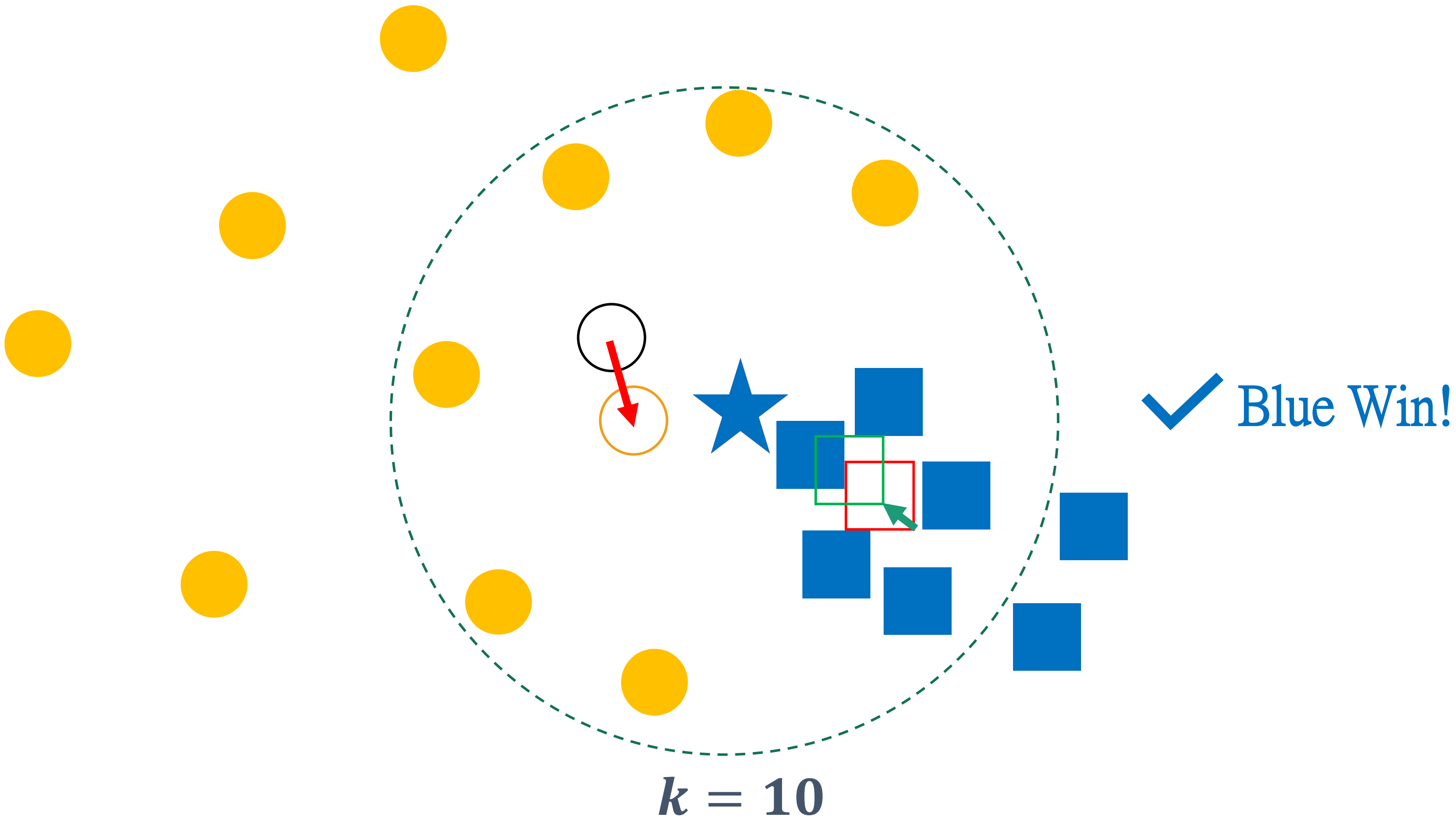
Yellow Class



$k = 10$

Centroid Displacement

Find the Minimum Displacement



Ensemble Condition

- In what situations would we need to use k-NN and CDNN?
  - High Confidence?
  - Low Confidence?

Confidence Sample

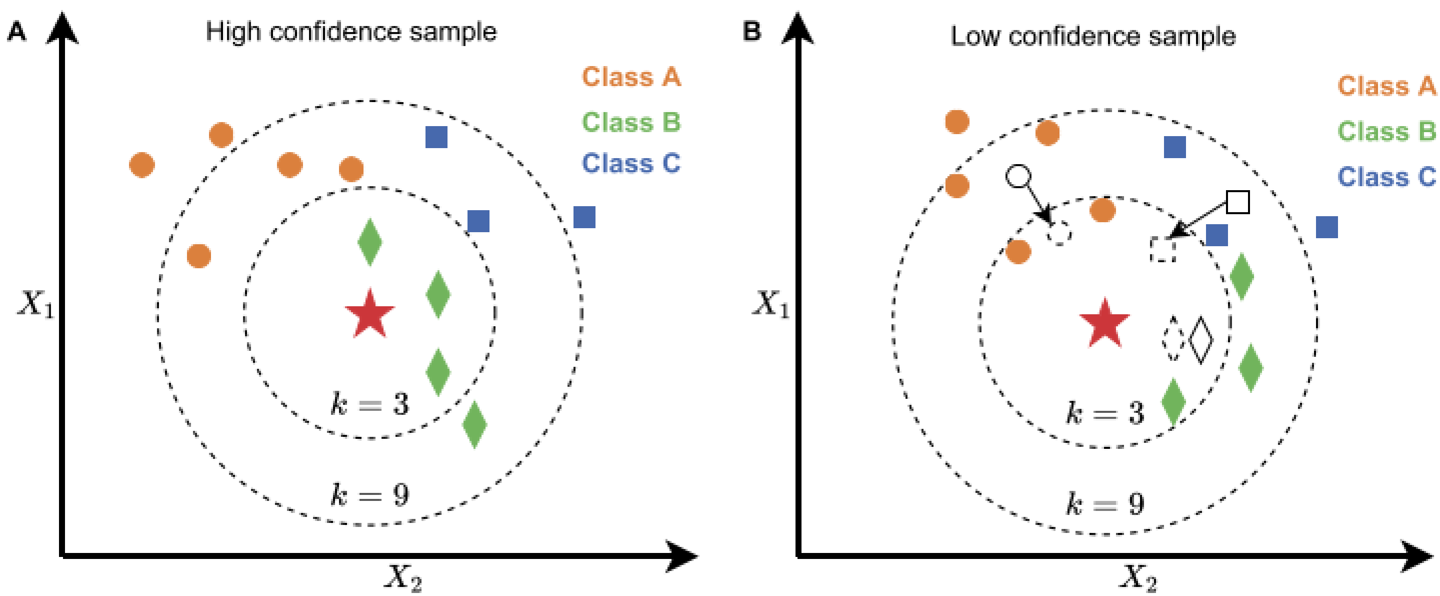


Fig. 1. Visual illustration of the proposed algorithm.

Algorithm

Input

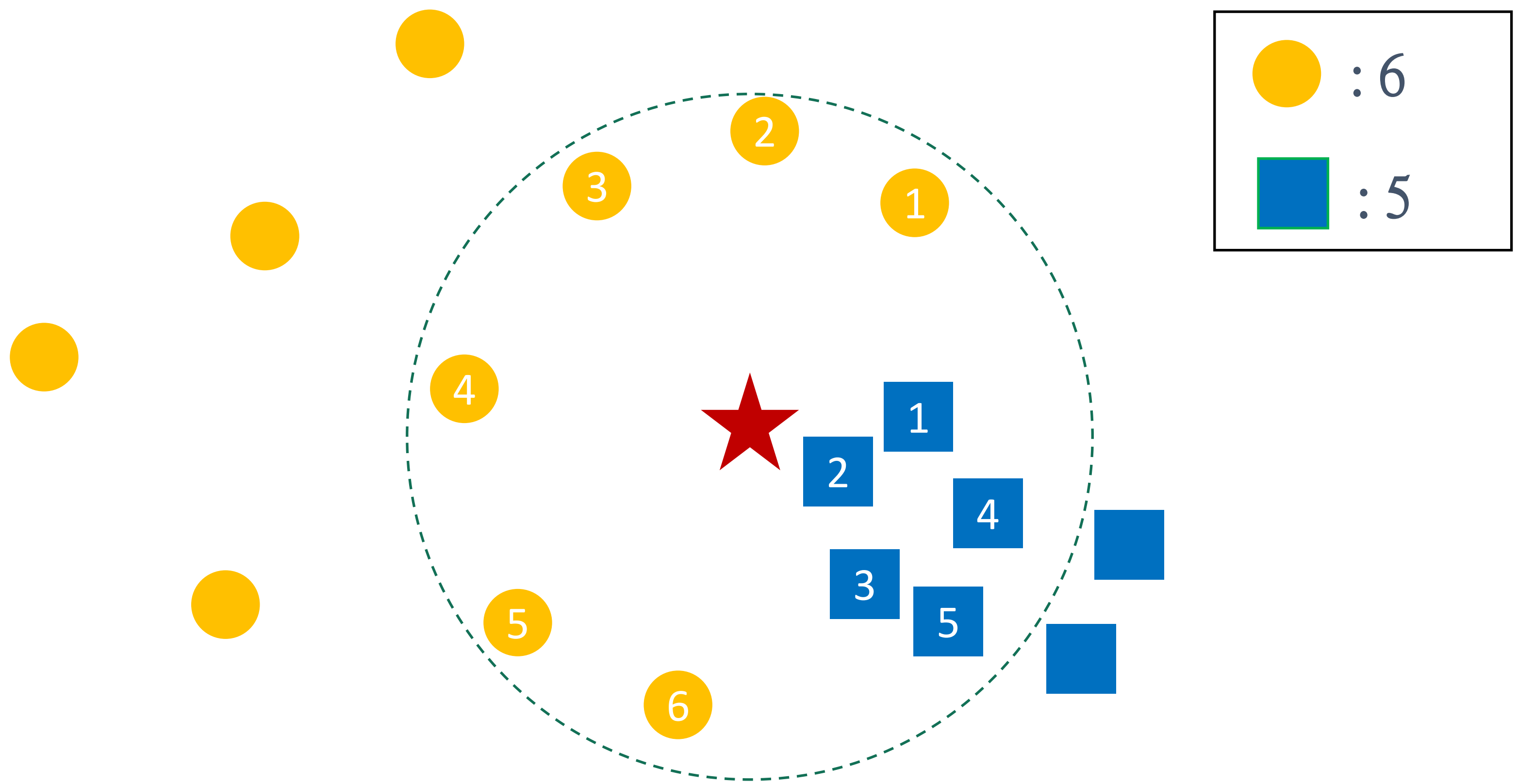
- Training Dataset
- Test instance: x
- Number of nearest neighbors: k

Centroid Displacement-based $\boldsymbol{k}$-NN (CDNN)

1. Compute the Euclidean distance between instance x and every points in dataset.
2. Find the k-NNs of instance x by Euclidean distance.

Centroid Displacement-based $\boldsymbol{k}$-NN (CDNN)

3. Count the number of clusters in k-NN statistically.

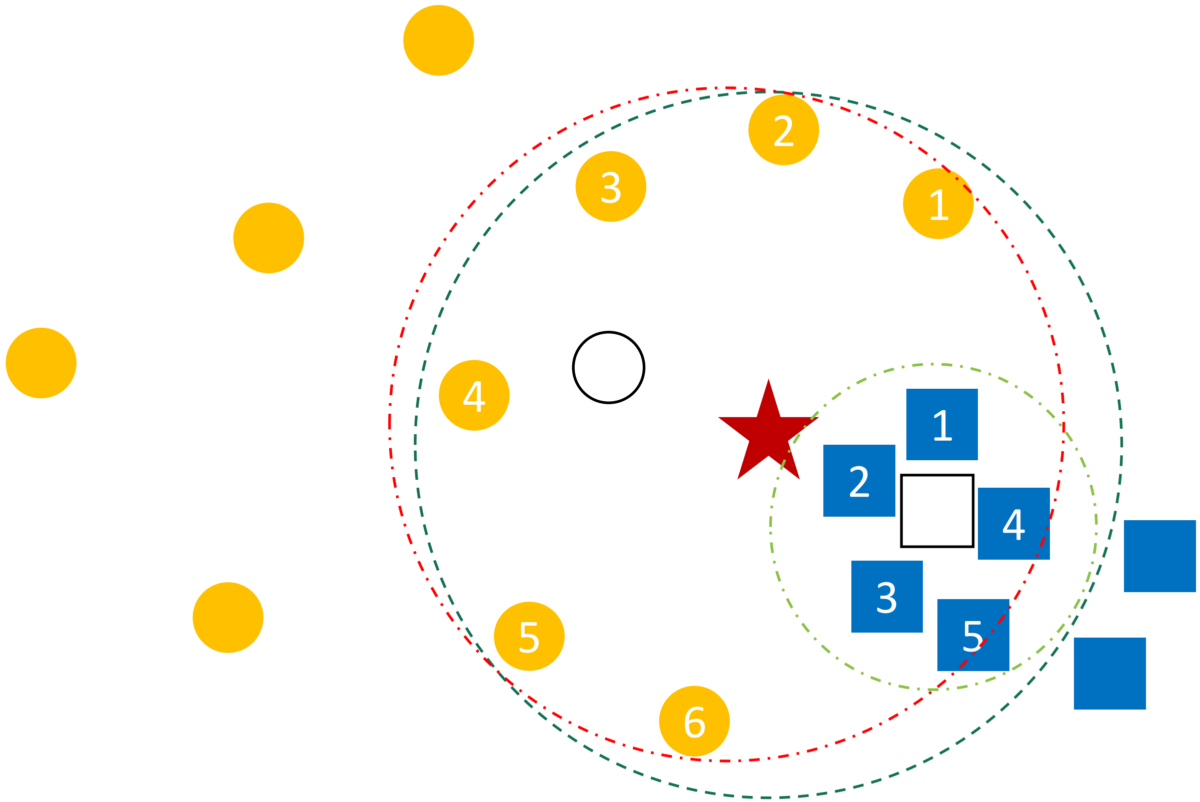


$\boldsymbol{k = 10}$

Centroid Displacement-based $\boldsymbol{k}$-NN (CDNN)

4. Find centroid of every clusters which exists in k-NN.

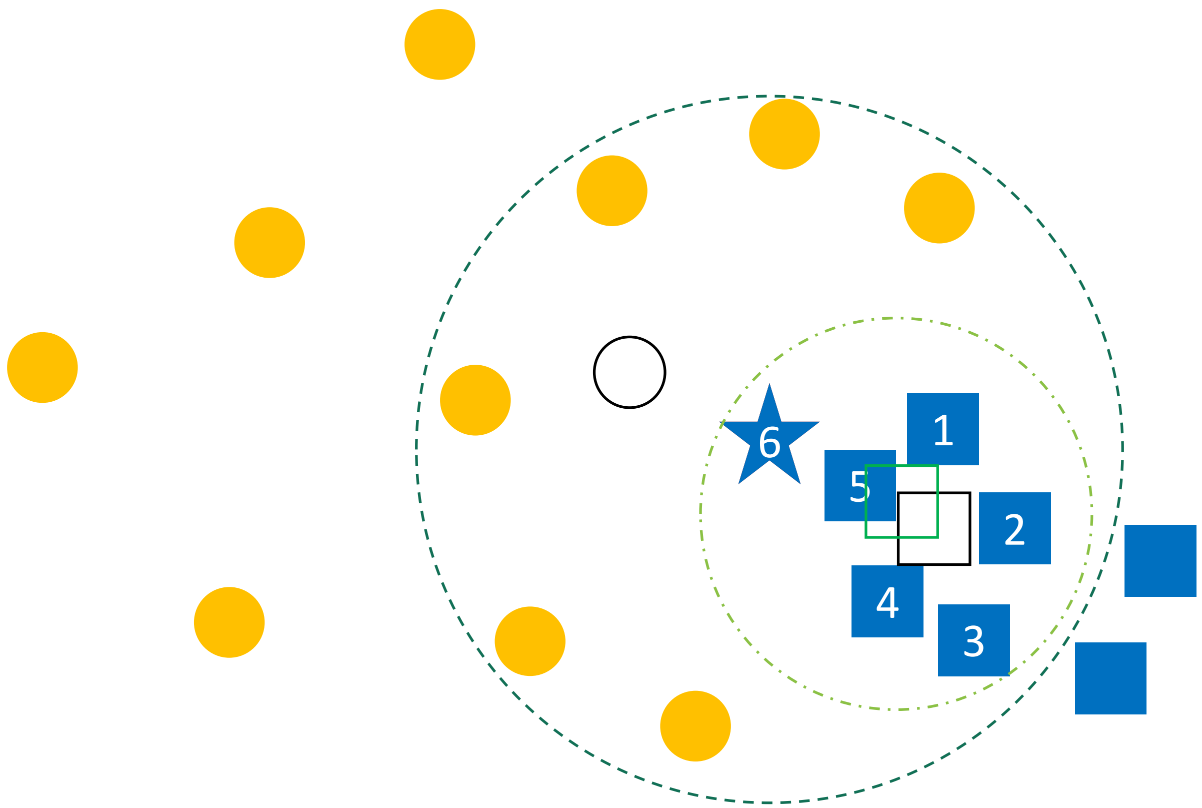
$$\mu_i = \frac{1}{n_i} \sum_{x_j \in C_i} x_j$$



$\boldsymbol{k = 10}$

Centroid Displacement-based $\boldsymbol{k}$-NN (CDNN)

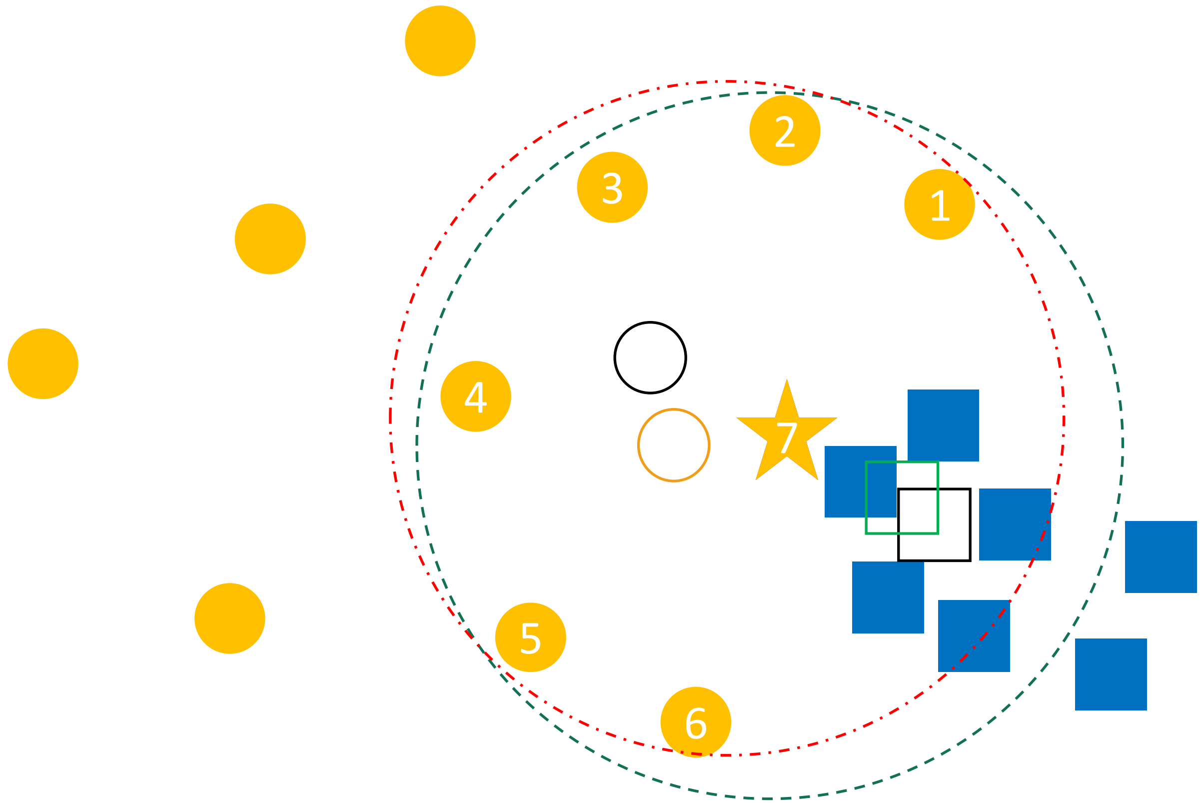
5. Re-compute centroid of every clusters after adding instance x.



$\boldsymbol{k = 10}$

Centroid Displacement-based $\boldsymbol{k}$-NN (CDNN)

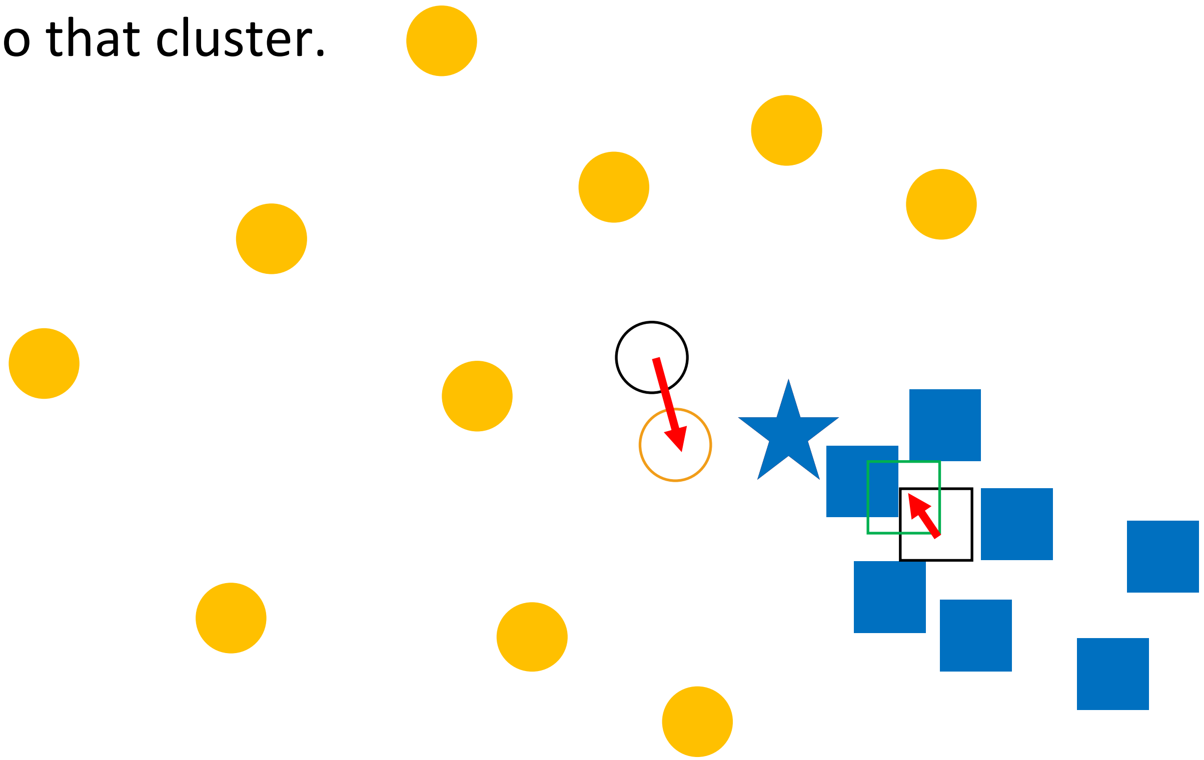
5. Re-compute centroid of every clusters after adding instance x.



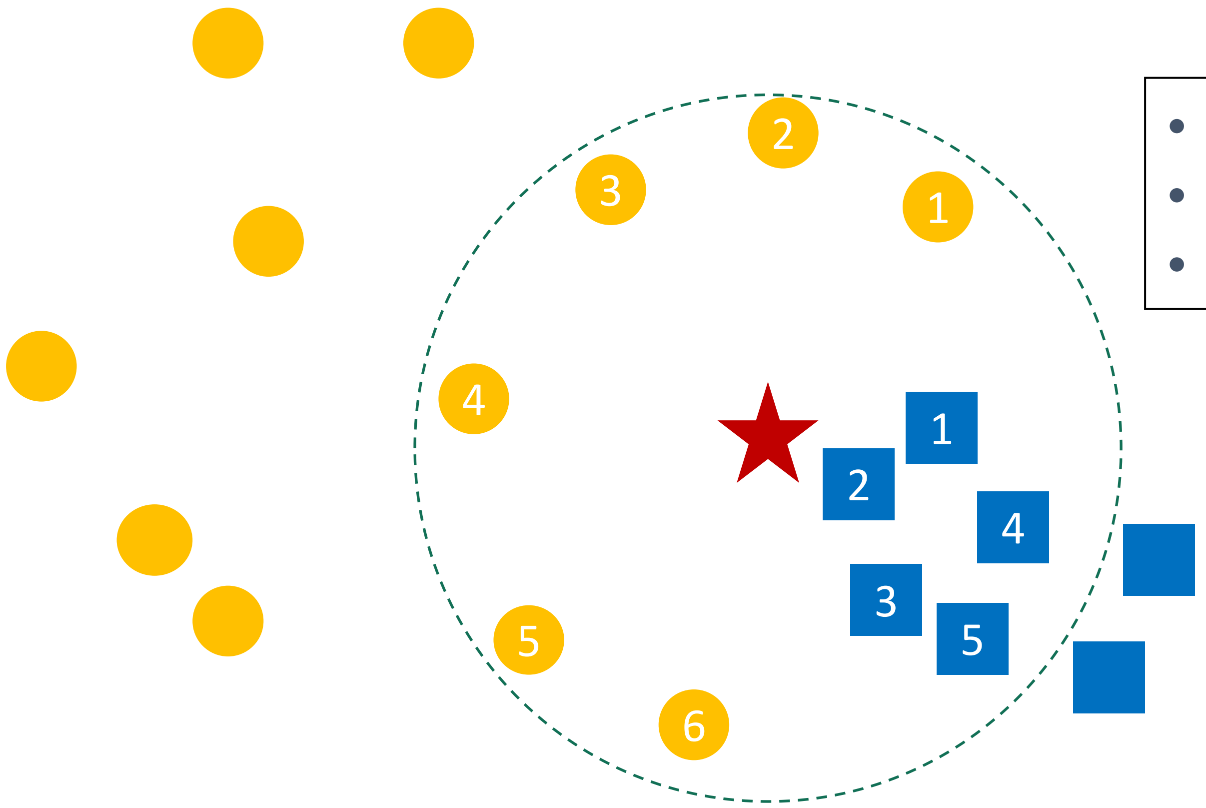
$\boldsymbol{k = 10}$

Centroid Displacement-based $\boldsymbol{k}$-NN (CDNN)

6. Find the **minimum displacement** in clusters and assign instance x into that cluster.



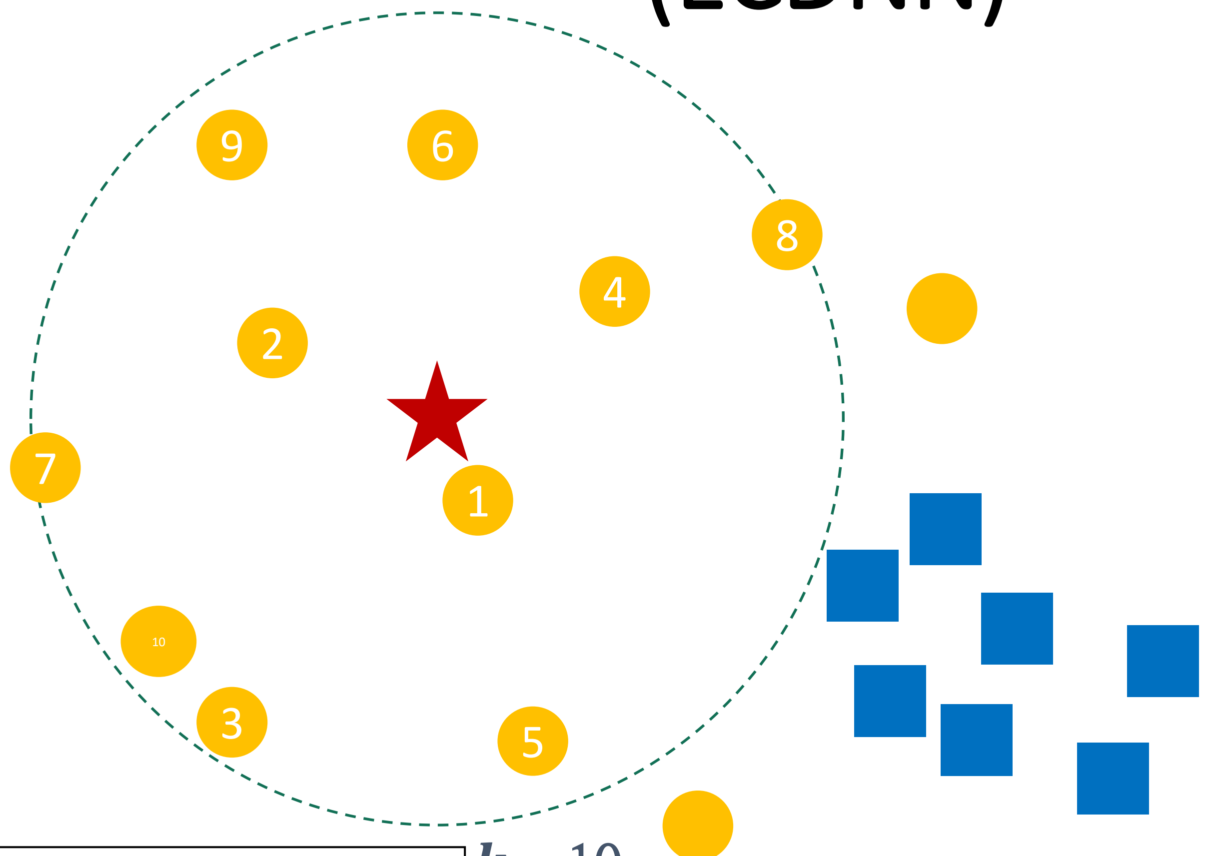
Ensemble Centroid Displacement-based $\boldsymbol{k}$-NN (ECDNN)



- Not the same
- Low Confidence
- CDNN

$\boldsymbol{k = 10}$

Ensemble Centroid Displacement-based k-NN (ECDNN)



k =10

- The same
- High Confidence
- k-NN

Time Complexity - $\boldsymbol{O(n^2)}$

1. Compute the Euclidean distance between instance x and every points in dataset and Find their k-NNs .$O(n^2)$
2. Count the number of clusters in k-NN statistically. $O(k)$
3. Find centroid of every clusters which exists in k-NN. $O(k)$
4. Re-compute the centroid of every clusters after adding instance x. $O(k)$
5. Find minimum displacement in clusters and assign instance x into that cluster. $O(k)$

Experiment

Comparing Algorithm

- k-NN Related Classification
  - k-NN
  - WKNN (weighted KNN)
    - It takes the **weight of the feature** index into account, will contribute to the improvement of the classification performance.
  - Radius NN (RNN)
    - It finds its all the neighbors **within a given radius r**.
  - NC (Nearest Centroid neighbors)
    - NC assigns instance x to the class of training sample whose local mean is closet to it.
  - CDNN
  - ECDNN

Comparing Algorithm

- Parameter
  - $k = [5, 25]$, increment by 2
  - Radius used the default parameters
- Experiment
  - Repeated 10 times, with each using 5-fold crossing validation per dataset
  - Total 50 results were collected and averaged.

Datasets Evaluation Indies

- $\#C$ is number of clusters
- IR is imbalance ratio
  - The ratio of the sample size of the largest majority class and that of the smallest minority class
- $\#S$ is sample size (number of points)
- $\#F$ is number of features (dimension)

Experiment Evaluation Index

- **Macro-F1**
  - [0, 1], the **higher is the better**
  - It the computation of the **harmonic mean** of *Recall* and *Precision*

$$F1 - score_i = 2 * \frac{Precision * Recall}{Precision + Recall}$$

$$Macro - F1 = \frac{F1 - score_1 + F1 - score_2 + \cdots + F1 - score_c}{c}$$

  - c is the number of classes

Datasets

(6 Synthetic + 16 Real-World)

Name	Source	$\#C$	IR	$\#S$	$\#F$	$\#F/\#S$	$\#F_{PCA}/\#S$	$\#F/\#F_{PCA}$	Sparsity
iris	UCI	3	-	150	4	0.0267	0.0133	2.00	0.0000
wine	UCI	3	-	178	13	0.0356	0.0223	1.60	0.4971
breastcancer	UCI	2	1.7:1	569	30	0.0527	0.0176	3.00	0.0047
digits	UCI	10	-	1797	64	0.0730	0.0562	1.30	0.0000
olivetti	UCI	40	-	4096	400	10.2400	0.3075	33.30	0.0000
paris	UCI	2	1:1	5828	2200	2.6491	0.1314	20.17	0.0001
S1	Synthetic	2	1:1	100	2	0.0200	0.0200	1.00	0.0000
S2	Synthetic	2	1:1	1000	2	0.0020	0.002	1.00	0.0000
S3	Synthetic	3	-	1000	10	0.0100	0.0090	1.11	0.0000
S4	Synthetic	3	-	1000	50	0.0500	0.0420	1.19	0.0000
S5	Synthetic	10	-	5000	50	0.0100	0.0092	1.09	0.0000
S6	Synthetic	10	-	5000	100	0.0200	0.0182	1.10	0.0000
ecoli	UCI	2	8.6:1	336	7	0.0208	0.0179	1.17	0.0020
optical_digits	UCI	2	9.1:1	5620	64	0.0114	0.0075	1.52	0.4961
satimage	UCI	2	9.3:1	6435	36	0.0056	0.0009	6.00	0.0000
pen_digits	UCI	2	9.4:1	10992	16	0.0015	0.0009	1.60	0.1369
abalone	UCI	2	9.7:1	4177	10	0.0024	0.0010	2.50	0.2223
sick_euthyroid	UCI	2	9.8:1	3163	42	0.0133	0.0057	2.33	0.4467
spectrometer	UCI	2	11:1	531	93	0.1751	0.0075	23.25	0.0000
car_eval_34	UCI	2	12:1	1728	21	0.0122	0.0087	1.40	0.7500
isolet	UCI	2	12:1	7797	617	0.0791	0.0260	3.04	0.0036
us_crime	UCI	2	12:1	1994	100	0.0502	0.0176	2.86	0.0560

Result

dataset	k-NN		WKNN		Radius NN		NC		CDNN		ECDNN	
	F1	rank	F1	rank	F1	rank	F1	rank	F1	rank	F1	rank
iris	0.9513	4	0.9586	2	0.9406	5	0.8574	6	**0.9599**	1	0.9518	3
wine	0.9681	3	0.9665	4	0.4705	6	0.9744	1	0.9721	2	0.9664	5
breastcancer	0.9571	4	0.9609	3	0.6193	6	0.9246	5	0.9626	2	**0.9659**	1
digits	0.9637	4	0.9669	3	0.1815	6	0.8889	5	0.9704	2	**0.9763**	1
olivetti	0.6914	5	0.7089	4	0.0527	6	0.8775	1	0.8222	3	0.8771	2
paris	0.5946	4	0.5881	5	0.5977	3	0.5181	6	0.5993	2	**0.6049**	1
S1	0.8844	5	0.9019	3	**0.9095**	1	0.8243	6	0.9009	4	0.9074	2
S2	**0.8992**	1	0.8943	2	0.8781	5	0.8408	6	0.8936	3	0.8899	4
S3	0.979	4	0.9798	3	0.5319	6	0.8736	5	0.9814	2	**0.9837**	1
S4	0.9642	4	0.9659	2	0.4655	6	0.7729	5	**0.9686**	1	0.9655	3
S5	0.9026	4	0.9095	3	0.1802	6	0.5837	5	**0.9181**	1	0.9140	2
S6	0.9075	4	0.9154	3	0.1802	6	0.5644	5	**0.9252**	1	0.9177	2
Average	0.8886	4	0.8931	3	0.5006	6	0.7917	5	0.9062	2	**0.9101**	1
ecoli	0.7892	2	**0.7974**	1	0.7754	5	0.6926	6	0.7881	3	0.7816	4
optical_digits	0.9769	4	0.9773	3	0.4741	6	0.7652	5	0.9791	2	**0.9823**	1
satimage	0.8127	4	0.8149	3	0.6901	5	0.4727	6	0.8222	2	**0.8340**	1
pen_digits	0.9933	4	0.9939	3	0.9726	5	0.5860	6	0.9949	2	**0.9964**	1
abalone	0.5304	4	0.5337	3	0.4754	6	**0.5842**	1	0.5271	5	0.5497	2
sick_euthyroid	0.7134	4	0.7451	3	0.6123	5	0.5332	6	0.7583	2	**0.7718**	1
spectrometer	0.8524	4	0.859	3	0.4804	6	0.6304	5	0.8692	2	**0.8727**	1
car_eval_34	0.7099	5	0.7274	3	0.4798	6	0.7251	4	0.7547	2	**0.7640**	1
isolet	0.9140	3	0.9140	3	0.4800	6	0.6018	5	0.9156	2	**0.9173**	1
us_crime	0.6241	5	0.6302	4	0.4804	6	**0.6862**	1	0.6428	3	0.6548	2
Average	0.7916	4	0.7993	3	0.5920	6	0.6277	5	0.8052	2	**0.8125**	1

F1 Score

(Wine)

Complexity

Dataset	$\#C$	IR	$\#S$	$\#F$	$\frac{\#F}{\#S}$	$\frac{\#F_{PCA}}{\#S}$	$\frac{\#F}{\#F_{PCA}}$	Sparsity
Wine	3	-	178	13	0.0356	0.0223	1.60	0.4971

Algorithm	F1	Rank
K-NN	0.9681	3
WKNN	0.9665	4
Radius NN	0.4705	6
NC	0.9744	1
CDNN	0.9721	2
ECDNN	0.9664	5

F1 Score

(Olivetti)

A lot of clusters

Dataset	$\#C$	IR	$\#S$	$\#F$	$\frac{\#F}{\#S}$	$\frac{\#F_{PCA}}{\#S}$	$\frac{\#F}{\#F_{PCA}}$	Sparsity
Olivetti	40	-	4096	400	10.24	0.3075	33.30	0.0000

Algorithm	F1	Rank
K-NN	0.6914	5
WKNN	0.7089	4
Radius NN	0.0527	6
NC	0.8775	1
CDNN	0.8222	3
ECDNN	0.8771	2

F1 Score

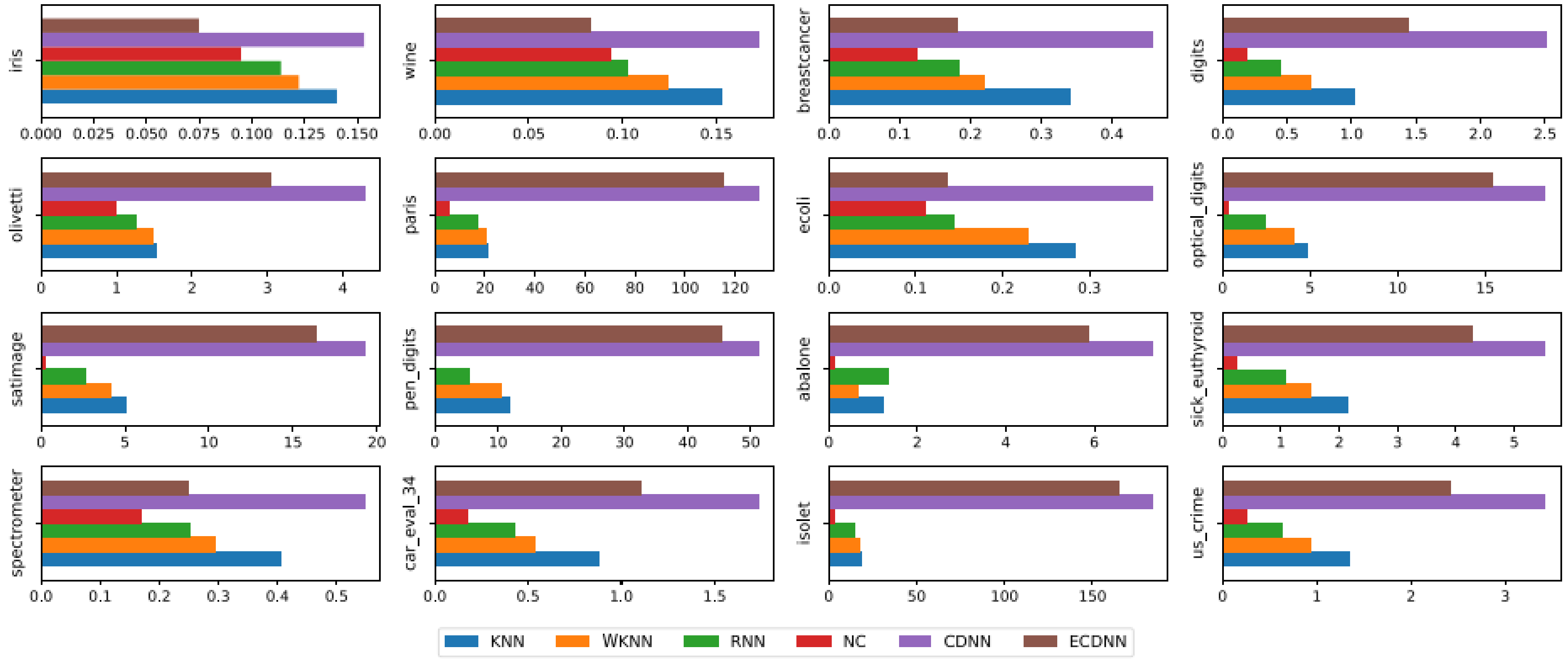
(car_eval_34)

Imbalance

Dataset	$\#C$	**IR**	$\#S$	$\#F$	$\frac{\#F}{\#S}$	$\frac{\#F_{PCA}}{\#S}$	$\frac{\#F}{\#F_{PCA}}$	**Sparsity**
Car_eval_34	2	12:1	1728	21	0.0122	0.0087	1.40	0.7500

Algorithm	**F1**	**Rank**
K-NN	0.7099	4
WKNN	0.7274	3
Radius NN	0.4798	5
NC	0.7251	6
CDNN	0.7547	2
ECDNN	0.7640	1

Execution Time



ECDNN is always faster than CDNN

My Experiment

My Experiment

- Dataset (downloaded from UCI)
  - Iris
  - Breastcancer
  - Satimage
  - Abalone
- Comparing Algorithm
  - ECDNN
  - CDNN
  - KNN
  - RNN

Environment

- Specification
  - Intel Core I7-8700 @ 3.20GHz
  - RAM: DDR4 16GB
  - Ubuntu 20.04
- Programming Language
  - C++ (ver. 9.4.0)

Parameter

- Let k set
  - [4, 26] increases by 1
- Let r set
  - [0.5, 30] increases by 0.5
- run $20\ X\ 5 - fold$ cross validation and take average

F1-Result

- Iris

Algorithm	Paper	My result
ECDNN	0.9518	0.9710 (14)
CDNN	**0.9599**	**09710 (14)**
KNN	0.9513	0.9690 (11)
RNN	0.9406	0.9501 (1.0)

- breastcancer

Algorithm	Paper	My result
ECDNN	**0.9659**	**0.9331 (17)**
CDNN	0.9626	0.9331 (17)
KNN	0.9571	0.9304 (10)
RNN	0.6193	0.9081 (30.0)

F1-Result

- satimage

Algorithm	Paper	My result
ECDNN	**0.8340**	**0.7698 (4)**
CDNN	0.8222	0.7698 (4)
KNN	0.8127	0.7626 (5)
RNN	0.6901	0.7167 (2.0)

- abalone

Algorithm	Paper	My result
ECDNN	**0.5497**	**0.5331 (24)**
CDNN	0.5271	0.5331 (24)
KNN	0.5304	0.5306 (5)
RNN	0.4754	0.4988 (0.5)

Time

- Iris

Algorithm	My result (ms)	My F1-score
ECDNN	257	0.9710 (14)
CDNN	272	**09710 (14)**
KNN	237	0.9690 (11)
RNN	**207**	0.9501 (1.0)

- breastcancer

Algorithm	My result (ms)	My F1-score
ECDNN	6453	**0.9331 (17)**
CDNN	6697	0.9331 (17)
KNN	**6242**	0.9304 (10)
RNN	6261	0.9081 (30.0)

Time

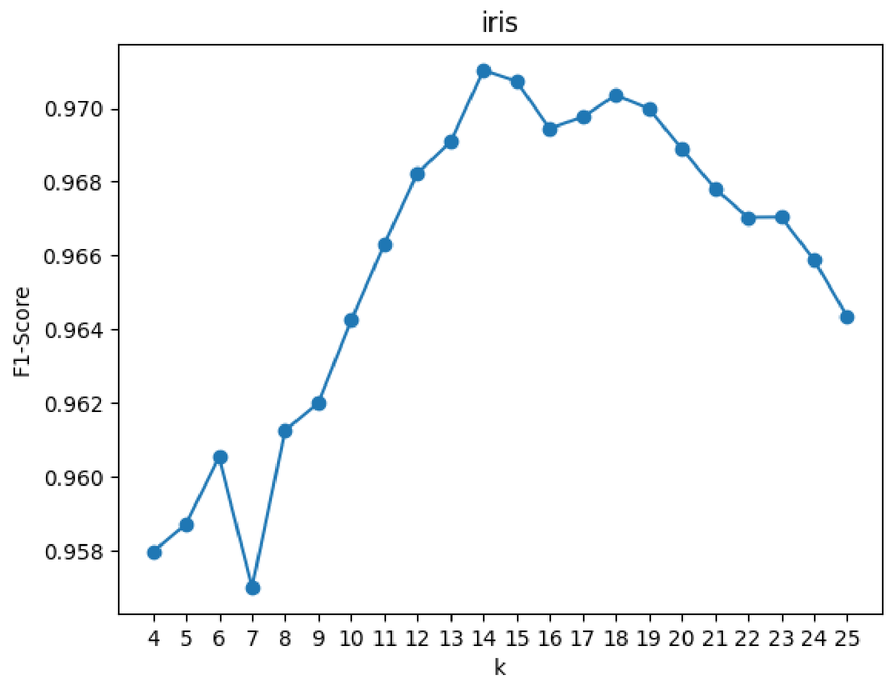
- satimage

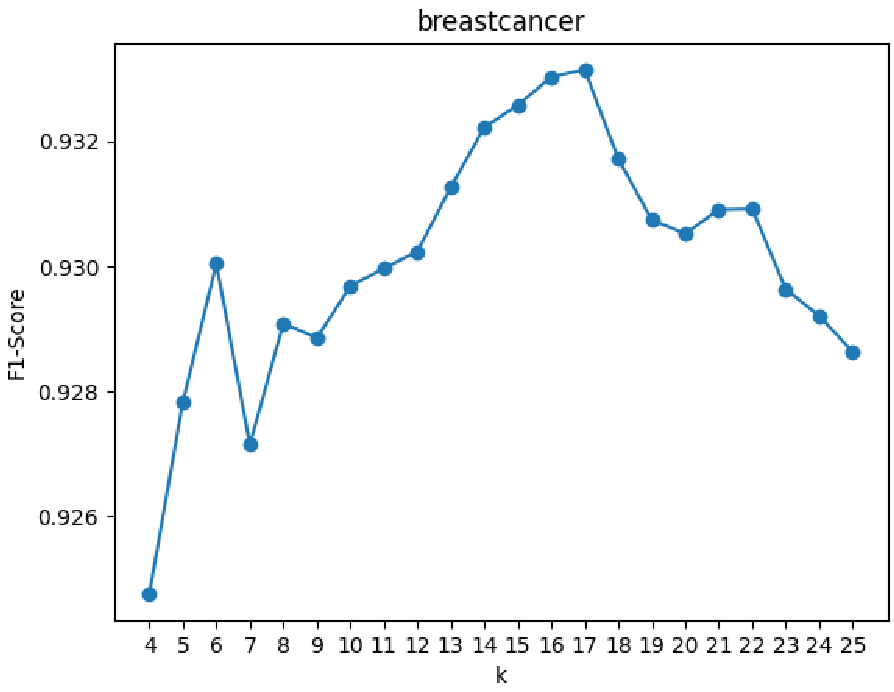
Algorithm	My result (ms)	My F1-score
ECDNN	759819	**0.7698 (4)**
CDNN	768572	0.7698 (4)
KNN	757313	0.7626 (5)
RNN	**748977**	0.7167 (2.0)

- abalone

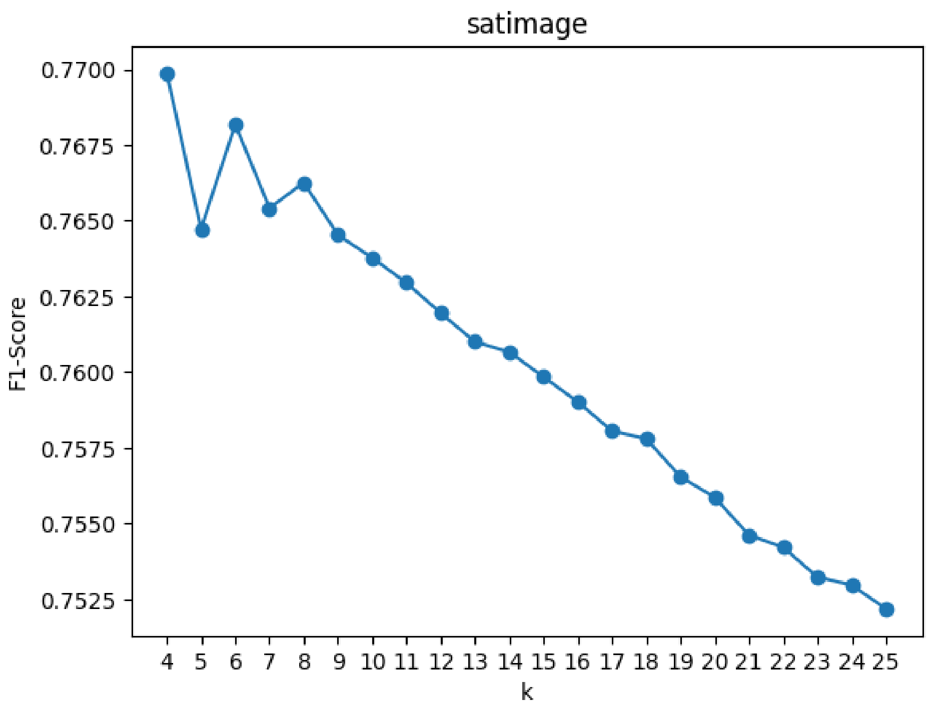
Algorithm	My result (ms)	My F1-score
ECDNN	134499	**0.5331 (24)**
CDNN	137200	0.5331 (24)
KNN	128028	0.5306 (5)
RNN	**126160**	0.4988 (0.5)

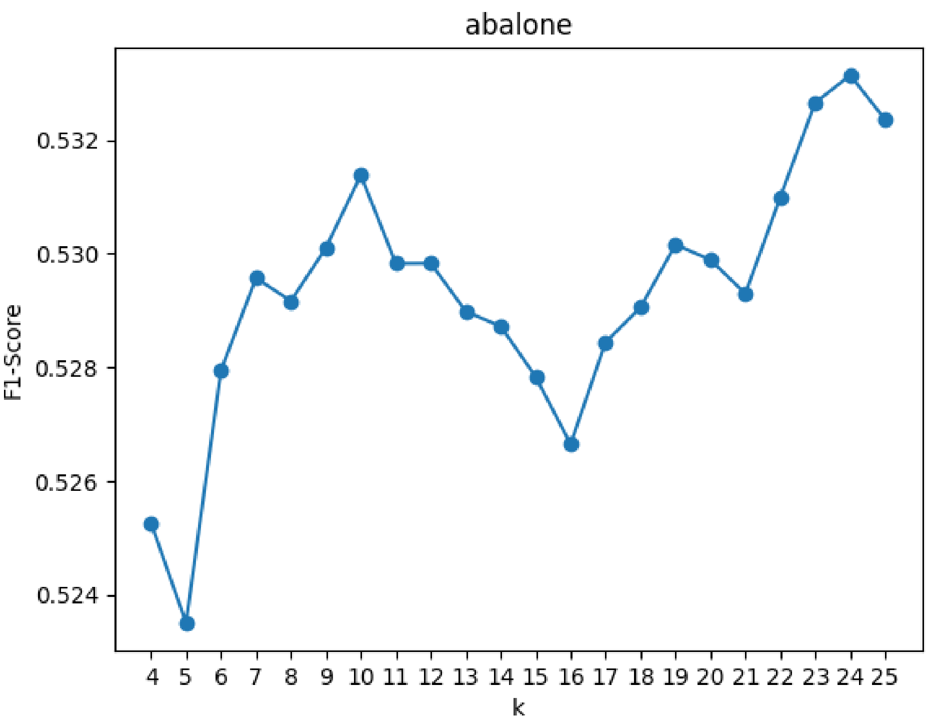
Varing k for ECDNN result





Varing k for ECDNN result





Conclusion

Conclusion

- CDNN is suitable for dataset with varying densities.
- CDNN is slower than k-NN, since we need to computer more information.
- ECDNN considers the confidence of every points in dataset, it avoids unnecessary computing for points with high confidence.
- ECDNN is faster than CDNN, it is more accuracy than k-NN.
- kNN-based classifications are sensitive for k-value.

Thank You for Your Time