Fakultas Informatika
School of Computing
Telkom University

Introduction to Artificial Intelligence– CII2M3

Nearest Neighbor

ADF – Start 08:30

Nearest Neighbor

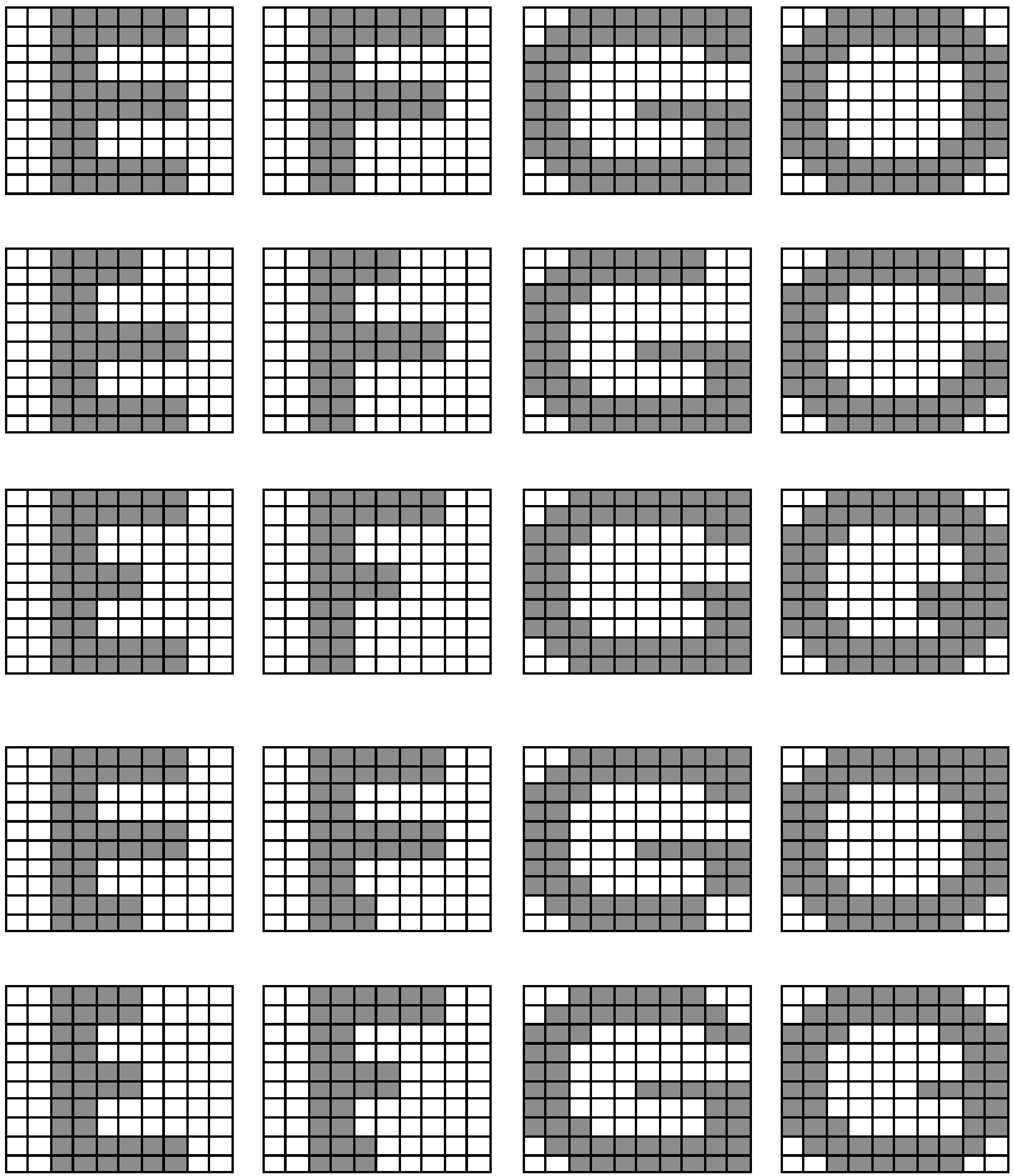
Enormous Attributes

- Decision tree is good for limited and discrete attributes
- In case of classification with numerous attributes,

  the tree resulted might be too complex and prone to overfit

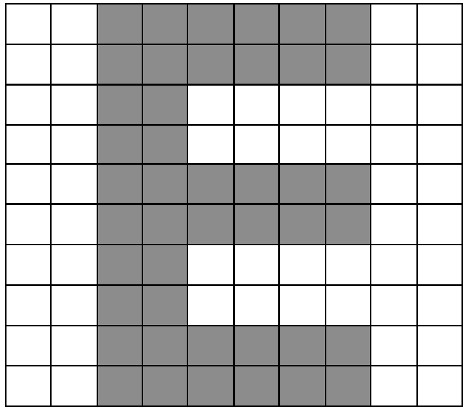
- For example: in image classification

Character Recognition



20 data
10x10 pixel
4 class

Character Recognition



0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0

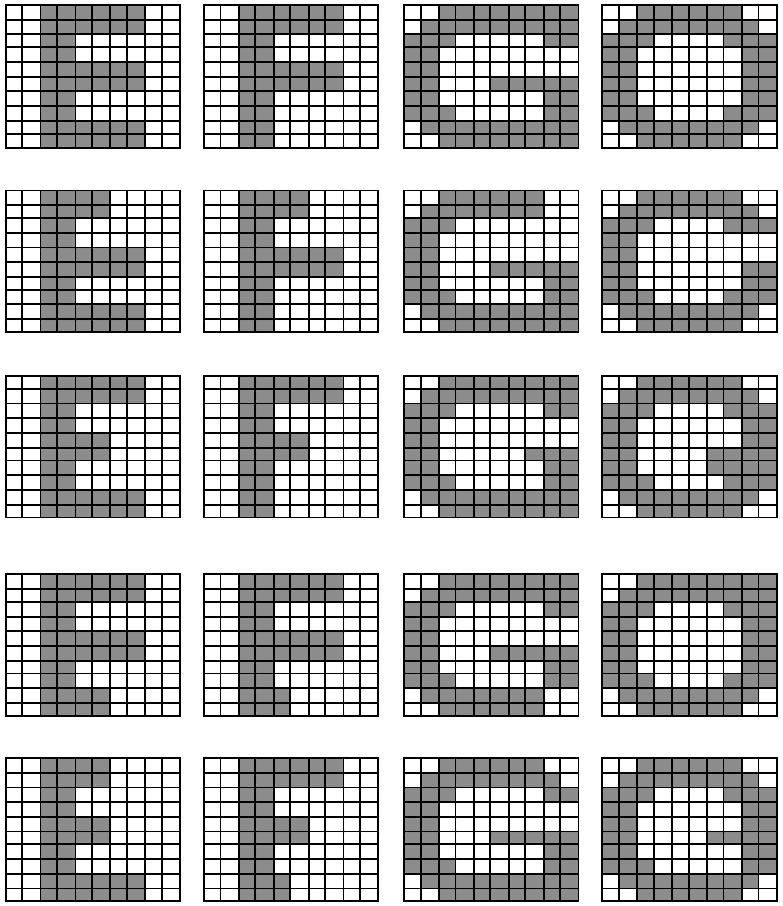
To ease the computation, change data into 1D array

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	1	1	1	1	1	1	0	0	0	0	1	1	1	.	.	.	.	.

...

91	92	93	94	95	96	97	98	99	100
0	0	1	1	1	1	1	1	0	0

Character Recognition Dataset



	Pix 1	Pix 2	Pix 3	Pix 4	Pix 5	…	Pix 100
E1	0	0	1	1	1	…	0
F1	0	0	1	1	1	…	0
G1	0	0	1	1	1	…	0
O1	0	0	1	1	1	…	0
..							
O5	0	0	1	1	1	…	0

20 data
100 dimensional
4 class

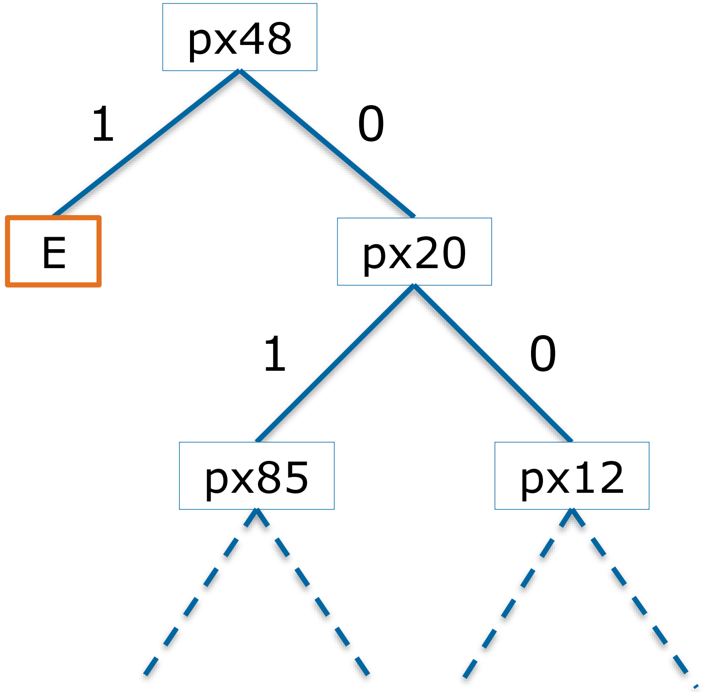
Character Recognition – ID3

- Let’s say we try to build character classifier based on pixel input using ID3
- What will happen?

Character Recognition – ID3

Character Recognition – ID3

- The tree will be too long
- A complete ID3 will result a 100-level-depth tree (100-dimensional input)
- Not efficient



Nearest Neighbor

Nearest Neighbor

- Simple algorithm that stores all available cases and classifies new cases based on a similarity measure
- Non-parametric techniques
- Other names:
  - Memory-Based Reasoning
  - Example-Based Reasoning
  - Instance-Based Learning
  - Case-Based Reasoning
  - Lazy Learning

Nearest Neighbor

- Training Process
  - Memorize all training data and labels
- Testing Process
  - Use **L1** or **L2** (or other distance choices) distance to each and every training data, return label of the closest one
  - L1 : Manhattan Distance
  - L2 : Euclidean Distance

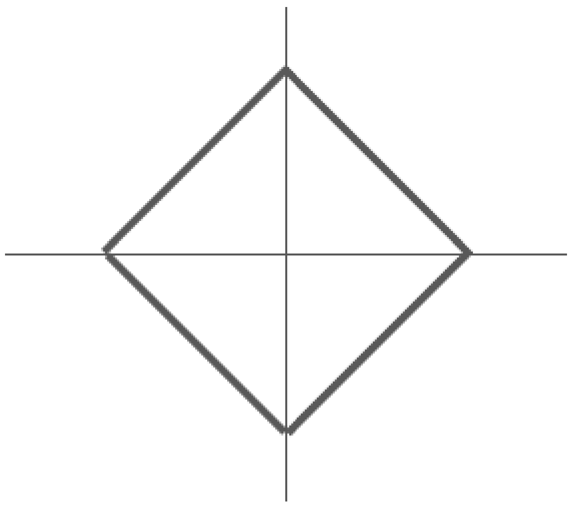
Distance Metric

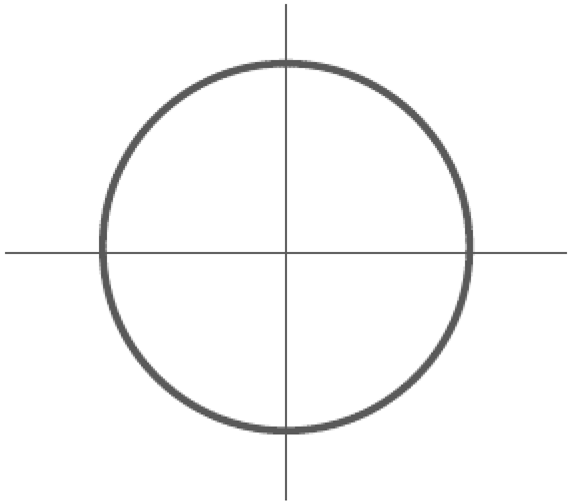
L1 (Manhattan) distance

$$d_1(x1, x2) = \sum_p |x1_p - x2_p|$$

L2 (Euclidean) distance

$$d_1(x1, x2) = \sqrt{\sum_p (x1_p - x2_p)^2}$$





Nearest Neighbor

L2 distance : $$d_1(x_1, y_2) = \sqrt{\sum_p \left(x_1^p - y_2^p\right)^2}$$

$$\left(\begin{array}{c} \text{test data } (x) \\ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \end{array} - \begin{array}{c} \text{train data } (y) \\ \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \end{array} \right)^2 = \begin{array}{c} \text{value differences} \\ \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \end{array} \xrightarrow{\text{sqrt(add)}} \sqrt{7}$$

Nearest Neighbor

L1 distance : $d_1(x_1, y_2) = \sum_p \left| x_1^p - y_2^p \right|$

$$\left| \overset{\text{test data } (x)}{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}} - \overset{\text{train data } (y)}{\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}} \right| = \overset{\text{value differences}}{\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}} \xrightarrow{\text{add}} \mathbf{7}$$

Nearest Neighbor

L1 distance : $d_1(x_1, y_2) = \sum_p \left| x_1^p - y_2^p \right|$

$$\left| \underset{\text{test data }(x)}{\begin{array}{|c|c|c|c|}\hline 1 & 1 & 1 & 1 \\ \hline 1 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 \\ \hline 1 & 0 & 0 & 0 \\ \hline \end{array}} - \underset{\text{train data }(y)}{\begin{array}{|c|c|c|c|}\hline 1 & 1 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 \\ \hline 1 & 1 & 1 & 1 \\ \hline 1 & 0 & 0 & 0 \\ \hline \end{array}} \right| = \underset{\text{value differences}}{\begin{array}{|c|c|c|c|}\hline 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 \\ \hline \end{array}} \xrightarrow{\text{add}} \mathbf{3}$$

Nearest Neighbor Algorithm

- For each test data A,
  - loop through all training data, calculate the distance
  - Choose the closest training data point X
  - Set the label of X as label of data test A

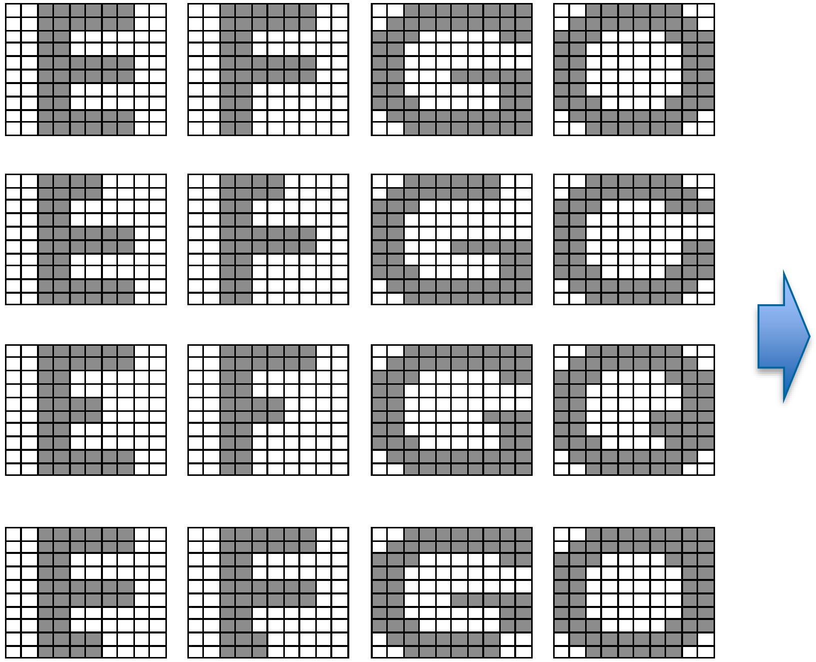
- Nearest neighbor intuition
  – Since A is similar to X (the attributes are close), then A and X must be the same class

Character Recognition with Nearest Neighbor

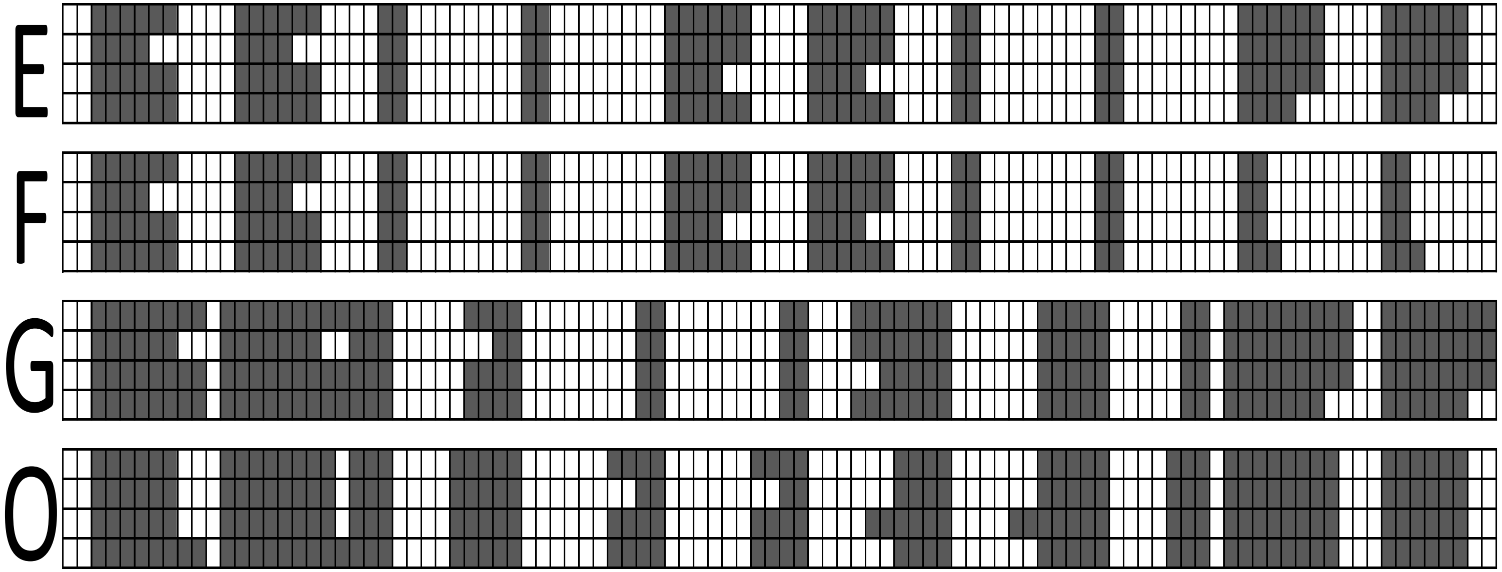
Training and Testing

Nearest Neighbor Learning Process

- Let’s say we use 16 data as training data

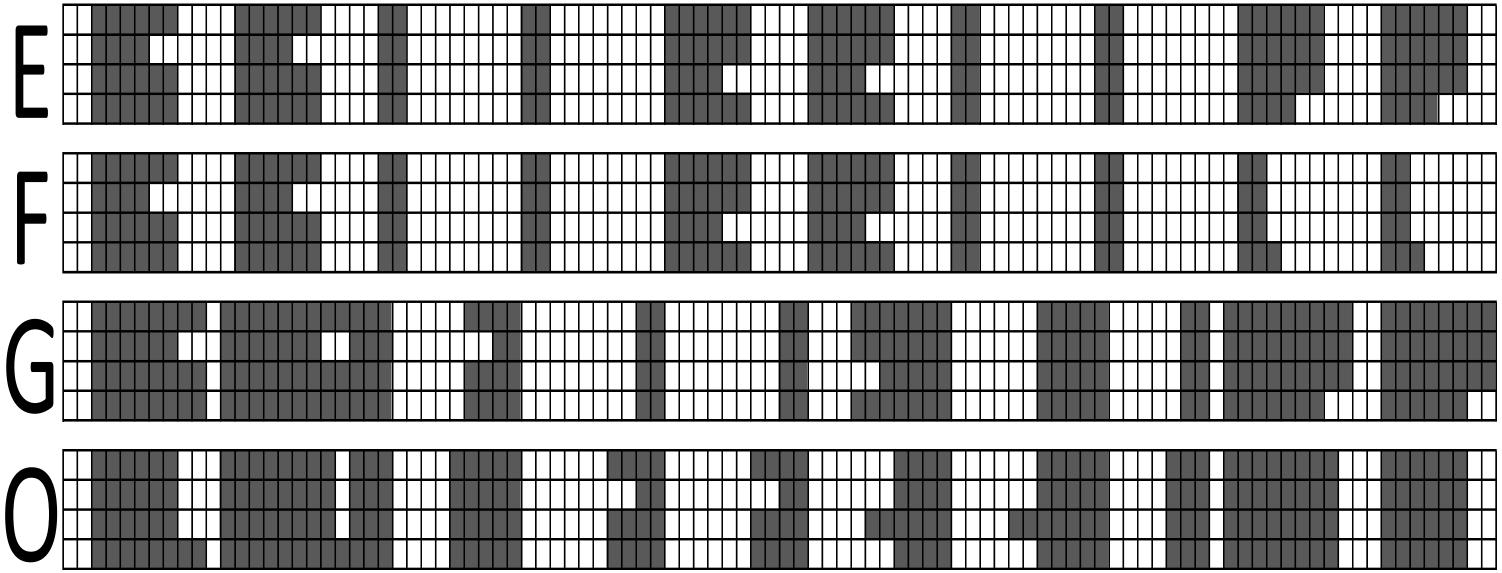


- Reshape into vectors



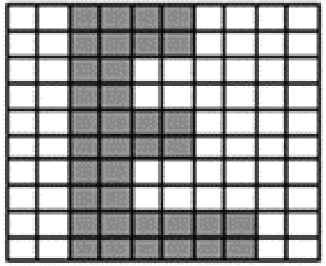
Nearest Neighbor Learning Process

- Save to database,
- And that’s it for Learning Process

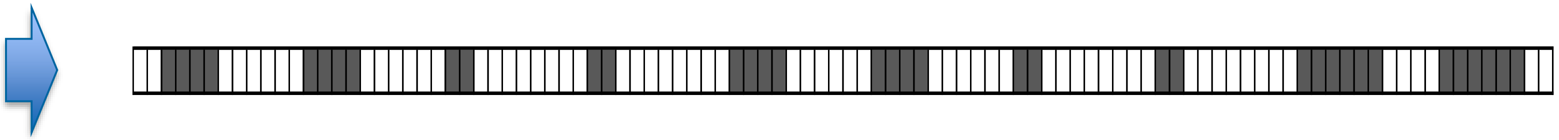


Nearest Neighbor Testing Process

- Now we want to classify new data using Nearest Neighbor

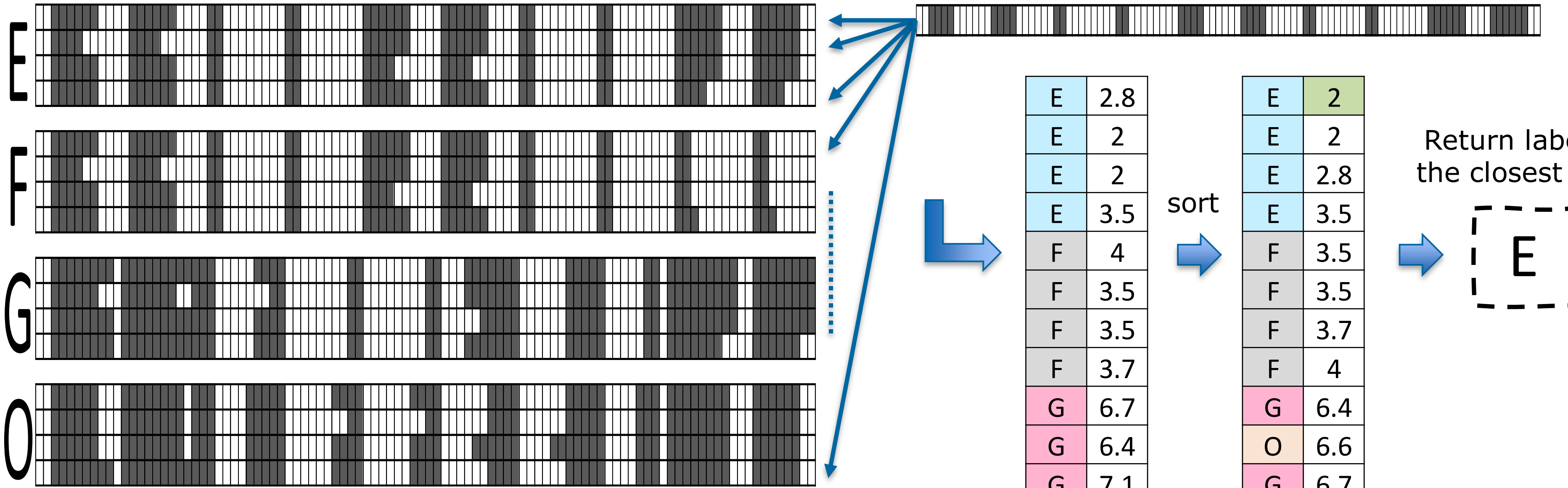


First, we change new data into 1 dimensional



Then calculate distance to the 16 data training

Nearest Neighbor Testing Process



Loop calculate the distance between test data and all training data resulting array of distance

E	2.8
E	2
E	2
E	3.5
F	4
F	3.5
F	3.5
F	3.7
G	6.7
G	6.4
G	7.1
G	6.9
O	6.9
O	6.6
O	7.2
O	7.1

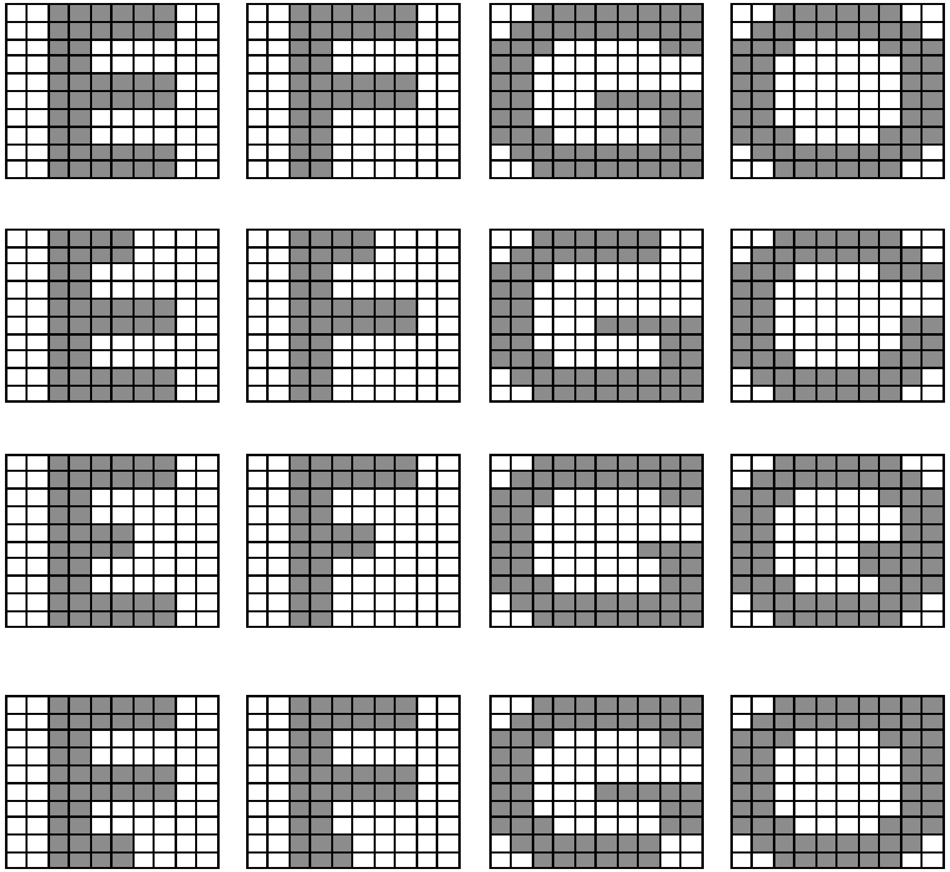
sort

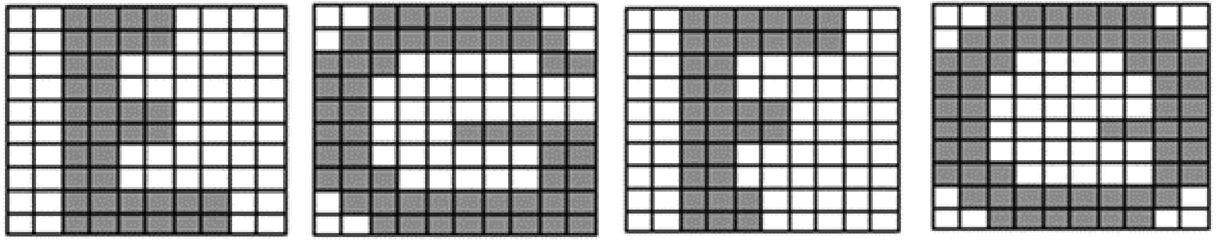
E	2
E	2
E	2.8
E	3.5
F	3.5
F	3.5
F	3.7
F	4
G	6.4
O	6.6
G	6.7
G	6.9
O	6.9
G	7.1
O	7.1
O	7.2

Return label of the closest data

E

Nearest Neighbor Testing Process





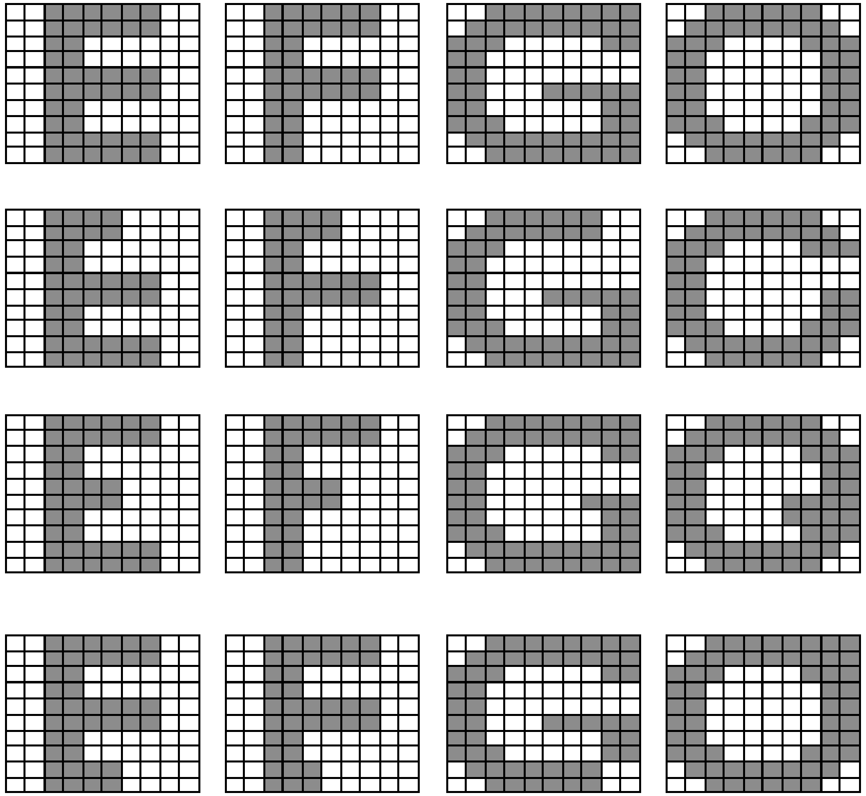
E	E	E	E	F	F	F	F	G	G	G	G	O	O	O	O
2.8	2	2	3.5	4	3.5	3.5	3.7	6.7	6.4	7.1	6.9	6.9	6.6	7.2	7.1
3.2	3.7	2.4	2.4	2.4	3.2	1.4	2	6.9	6.6	7.3	7	7.1	6.8	7.3	7.3
6.6	6.9	6.6	6.9	7.2	7.5	7.2	7.1	1.7	2.6	1.7	2.6	3.2	2.4	3.2	3.6
6.8	7.1	6.8	7.1	7.3	7.6	7.3	7.2	2.2	3	3.3	3	1.4	2.4	1.4	2.2

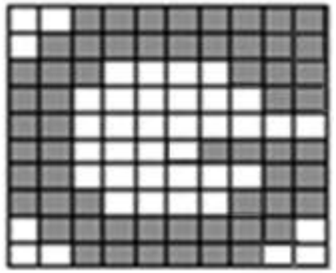
Nearest Neighbor Problem

Nearest Neighbor

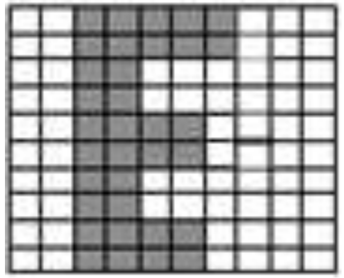
- Training Speed?
  - O(1)
  - No training, just save all training data
- Testing Speed?
  - O(N)
  - Linearly slower according to the size of training data

Character Recognition using Nearest Neighbor





G? O?



E? F?

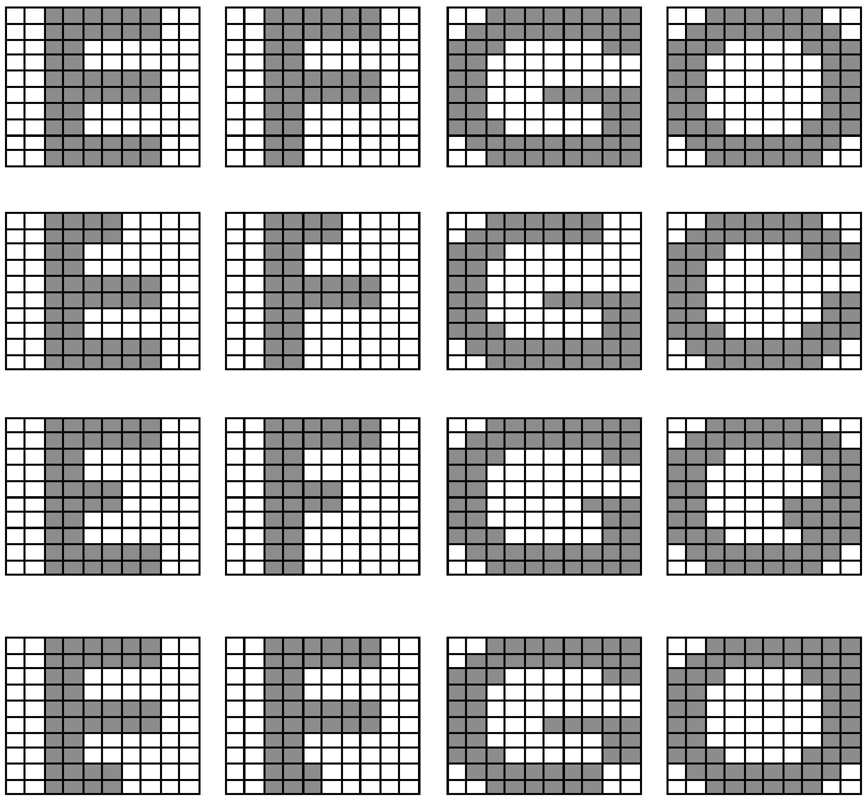
E	E	E	E	F	F	F	F	G	G	G	G	O	O	O	O
6.9	7.1	6.9	7.1	7.4	7.7	7.4	7.3	2.4	3.2	2.4	2	2.6	2.6	2.6	2
3.2	3.2	2.4	2.4	3.2	3.2	2.4	2.8	6.9	6.6	7.3	7	7.1	6.8	7.3	7.3

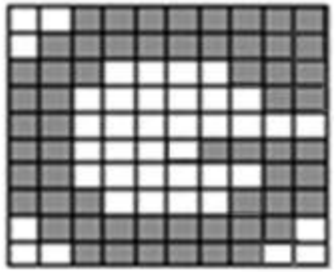
k-Nearest Neighbor

K-Nearest Neighbor

- Use many neighbors to determine the class
  - Choose k-closest data point
  - Majority vote the label
- K-NN can be use for Classification and Regression
  - For classification select the most frequent neighbor.
  - For regression calculate the average of K neighbors

Character Recognition using k-NN





K = 3

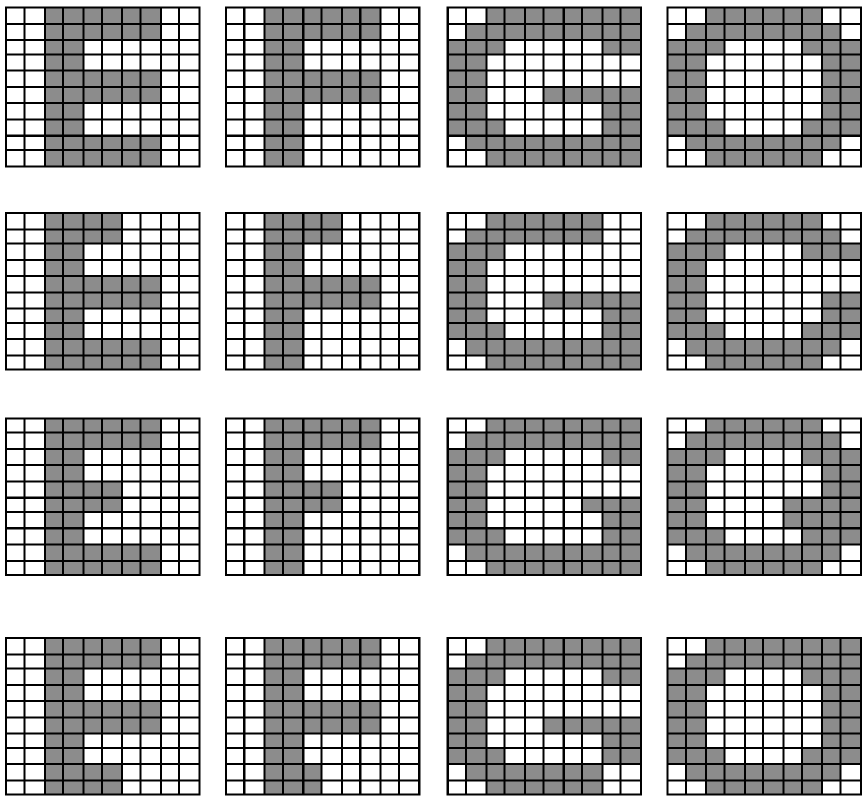
E	E	E	E	F	F	F	F	G	G	G	G	O	O	O	O
6.9	7.1	6.9	7.1	7.4	7.7	7.4	7.3	2.4	3.2	2.4	2	2.6	2.6	2.6	2

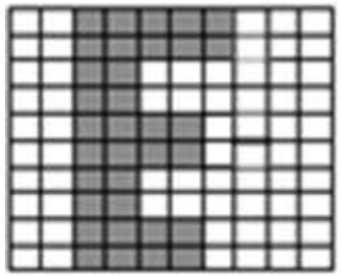
sort

G	O	G	G	O	O	O	G	E	E	E	E	F	F	F	F
2	2	2.4	2.4	2.6	2.6	2.6	3.2	6.9	6.9	7.1	7.1	7.3	7.4	7.4	7.7

2G, 1O = G

Character Recognition using k-NN





K = 3

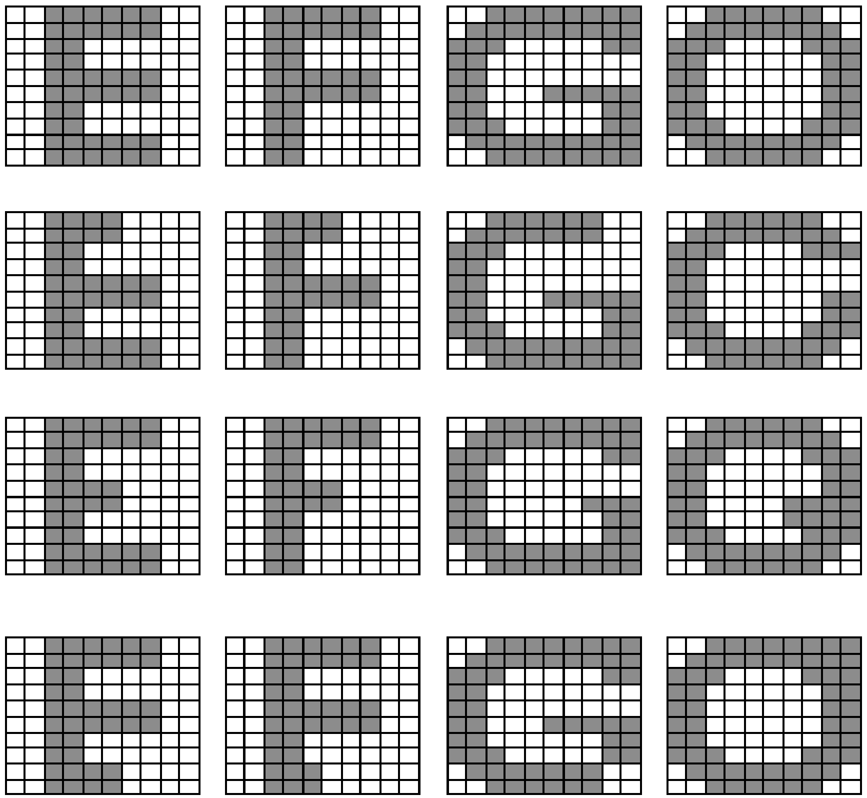
E	E	**E**	**E**	F	F	**F**	F	G	G	G	G	O	O	O	O
3.2	3.2	2.4	2.4	3.2	3.2	2.4	2.8	6.9	6.6	7.3	7	7.1	6.8	7.3	7.3

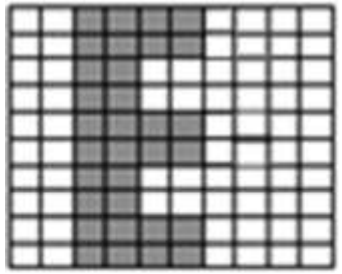
sort

E	**E**	**F**	F	E	E	F	F	G	O	G	G	O	G	O	O
2.4	2.4	2.4	2.8	3.2	3.2	3.2	3.2	6.6	6.8	6.9	7	7.1	7.3	7.3	7.3

2E, 1F = E

Character Recognition using k-NN



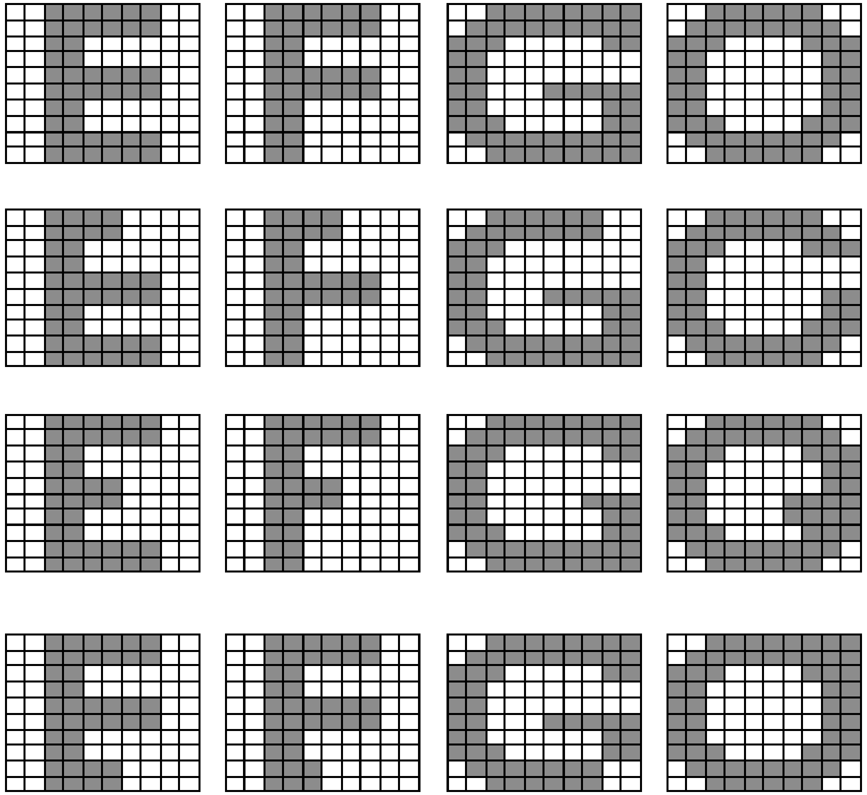


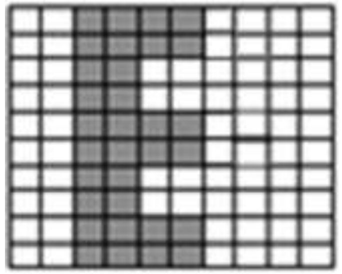
K = 3

E	E	E	E	F	F	F	F	G	G	G	G	O	O	O	O
3.5	2.8	2.8	2.8	3.5	2.8	2.8	3.2	7	6.7	7.4	7.1	7.2	6.9	7.5	7.4

E ?? F ??

Character Recognition using k-NN





K = 3

E ?? F ??

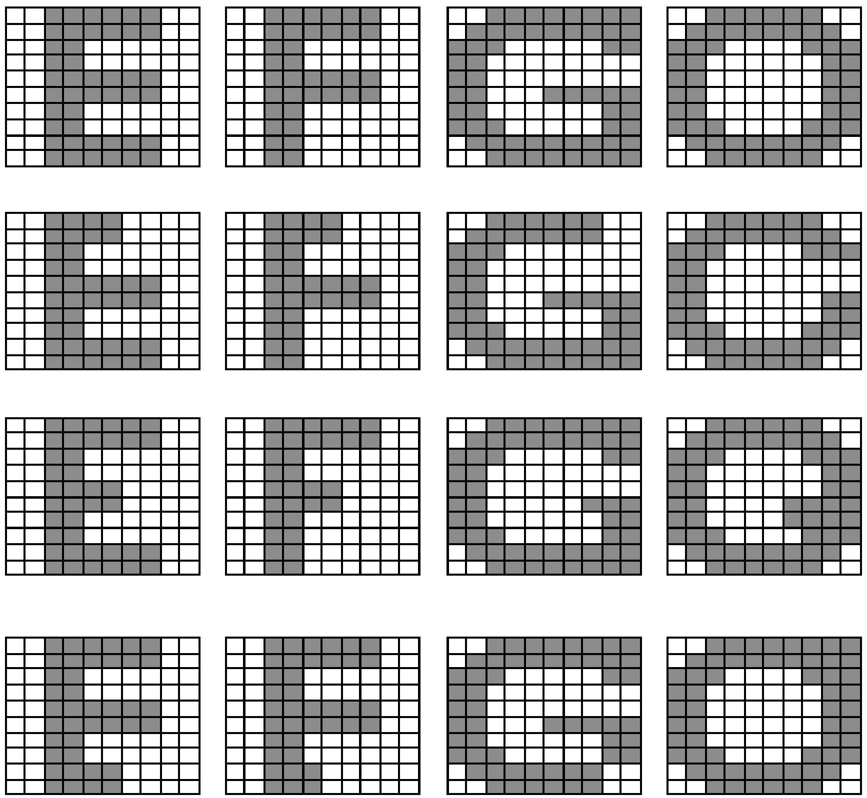
sort

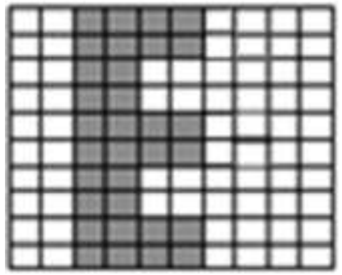
E	**F**	**E**	E	F	F	E	F	G	O	G	G	O	G	O	O
2.8	2.8	2.8	2.8	2.8	3.2	3.5	3.5	6.7	6.9	7	7.1	7.2	7.4	7.4	7.5

what will happened if the sorting result is

E	**F**	**F**	E	E	F	E	F	G	O	G	G	O	G	O	O
2.8	2.8	2.8	2.8	2.8	3.2	3.5	3.5	6.7	6.9	7	7.1	7.2	7.4	7.4	7.5

Character Recognition using k-NN





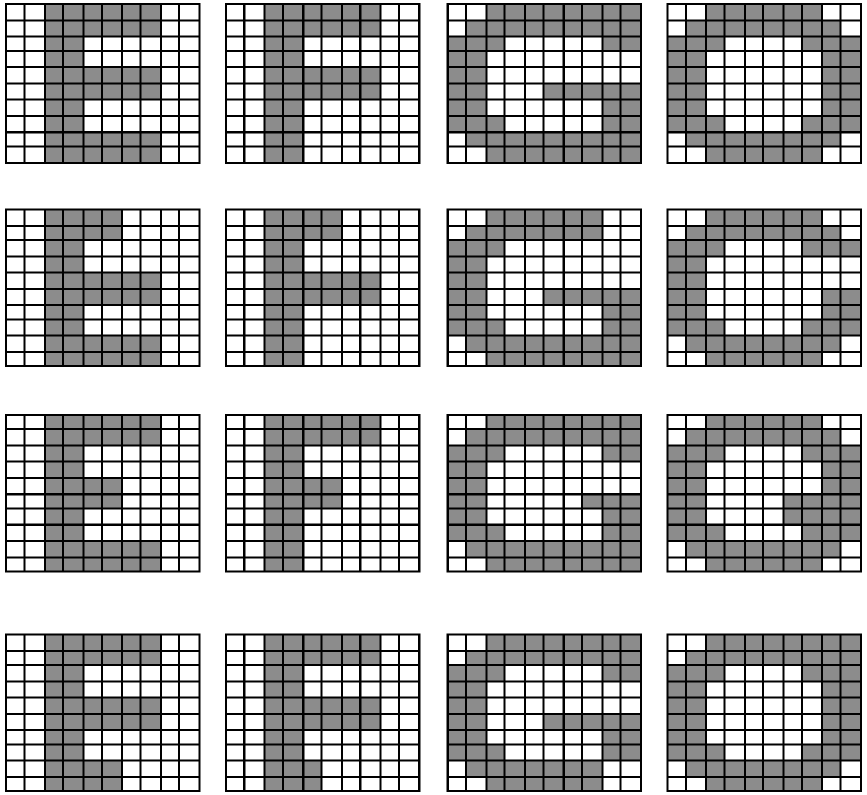
K = 3 Change → K = 5

sort

E	**F**	**E**	**E**	**F**	F	E	F	G	O	G	G	O	G	O	O
2.8	2.8	2.8	2.8	2.8	3.2	3.5	3.5	6.7	6.9	7	7.1	7.2	7.4	7.4	7.5

3E, 2F = E

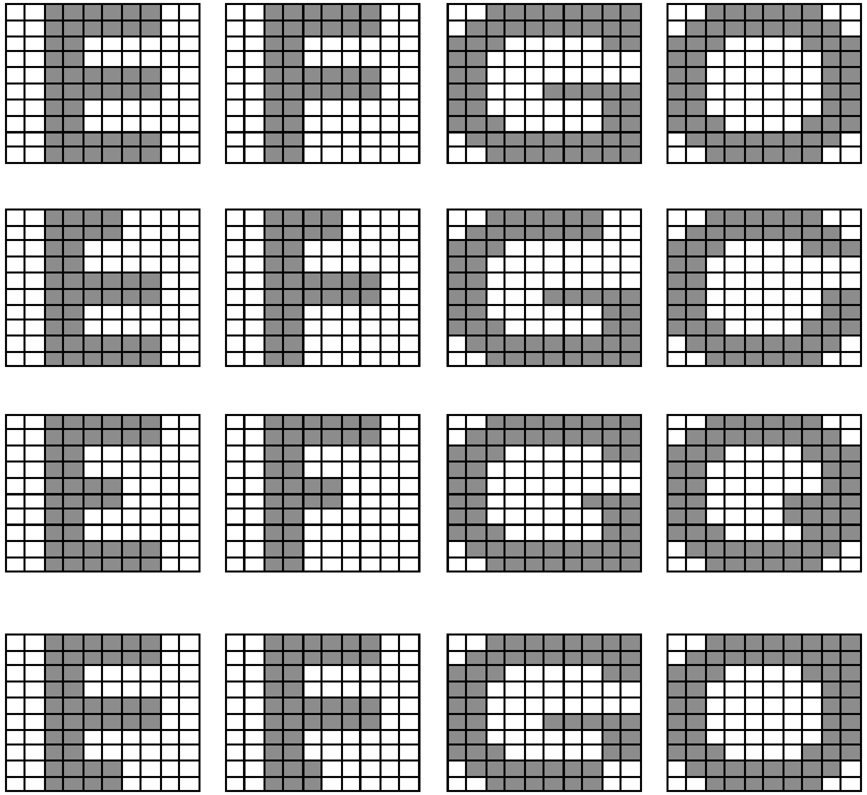
Character Recognition using k-NN



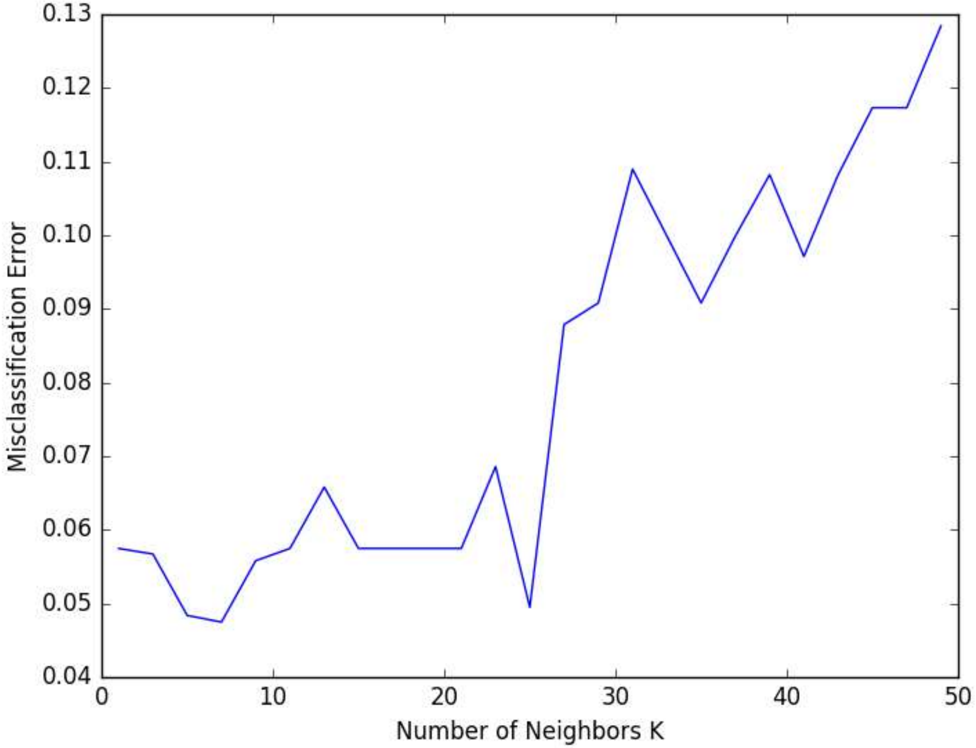
K = 3?
K = 5?
K = 7?
K = 10?

Which one is better?

Character Recognition using k-NN



Observe and determine k using validation set



k-Nearest Neighbors - Hyperparameters

- What is the best **distance** to use?
- What is the best value of **k** to use?

- Choices about the algorithm that we set rather than learn
- Very problem-dependent.
- Must try them all out and see what works best.
- Use train set and validation set to avoid overfitting

K-Fold Cross Validation

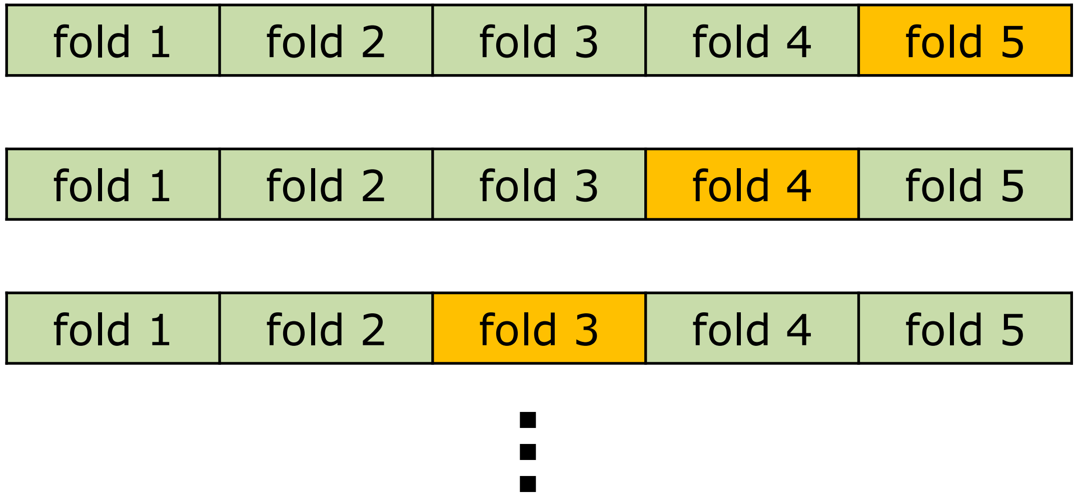
Start 09:30

Hyperparameter Observation on Trainset?

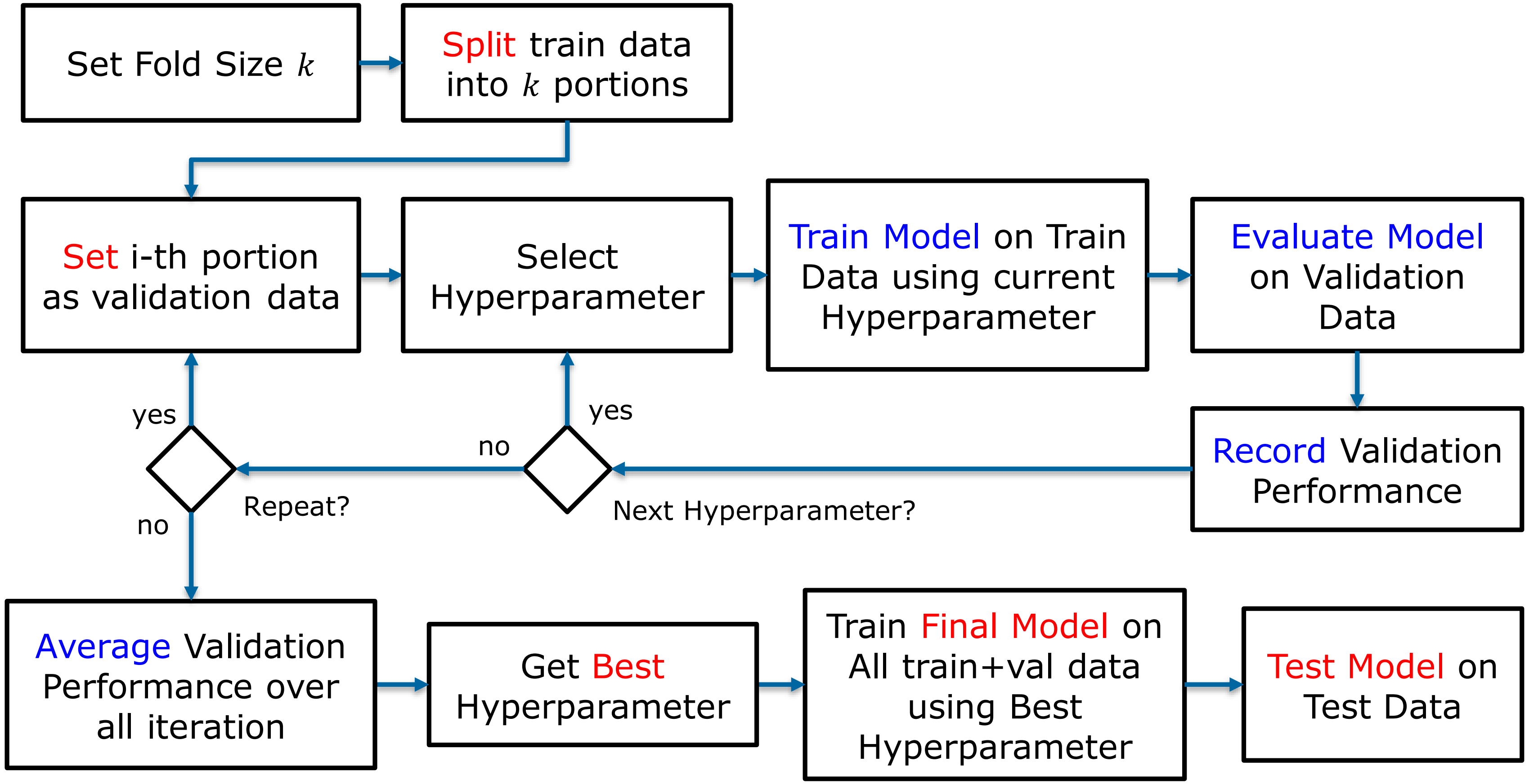
- We want to observe k={1, 2, 3, 4, 5}
- What would happen if we check the performance on train set?
- Best performance will ALWAYS be k=1

Example K-Fold Cross Validation

- Example K-Fold Cross Validation using 5 folds
  - Iteration 1
    - 1,2,3,4 train
    - Fold 5 validation
  - Iteration 2
    - 1,2,3,5 train
    - Fold 4 validation
  - Iteration 3
    - 1,2,4,5 train
    - Fold 3 validation
  - ...



K-Fold Val: Hyperparameter Observation



K-Fold Val: Hyperparameter Observation

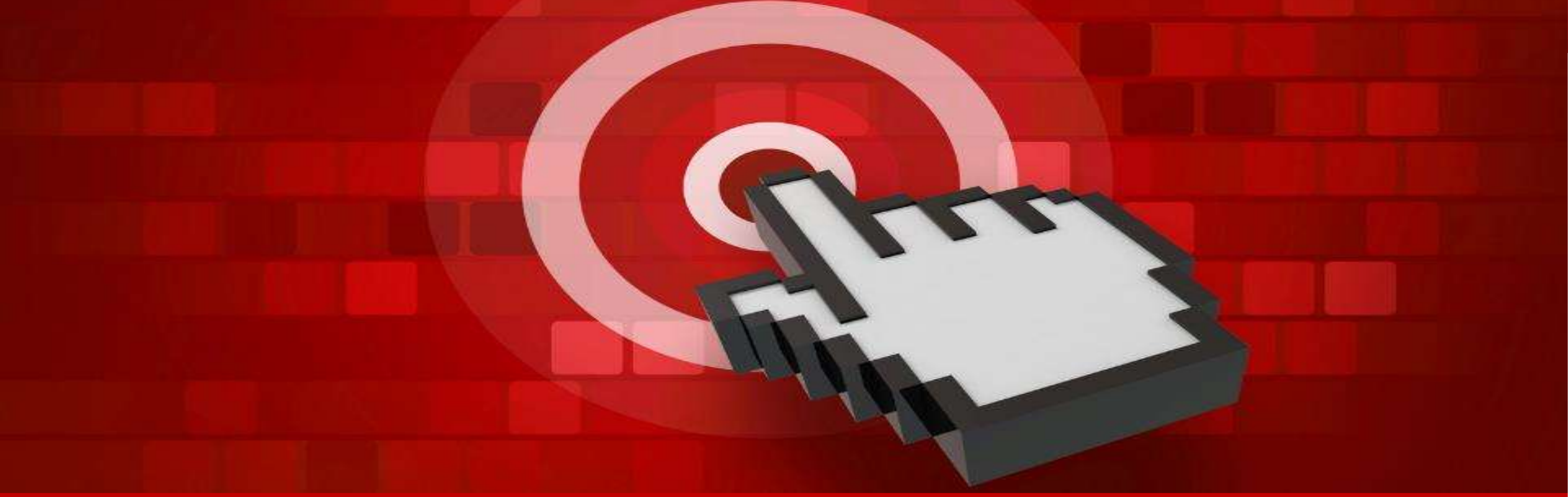
- For each iteration
  - Try all hyperparameters
  - Record the validation acc
- At the end of iteration, average the validation acc
- Choose best hyperparameter
- Use the chosen hyperparameter to test new data using ALL data (train+val)

	kNN validation				
Iteration	1	2	3	4	5

Note on kNN Hyperparameter Observation

- Myth: k value should be an odd-number

Question?



THANK YOU