K-Nearest Neighbour (K-NN) Classifier

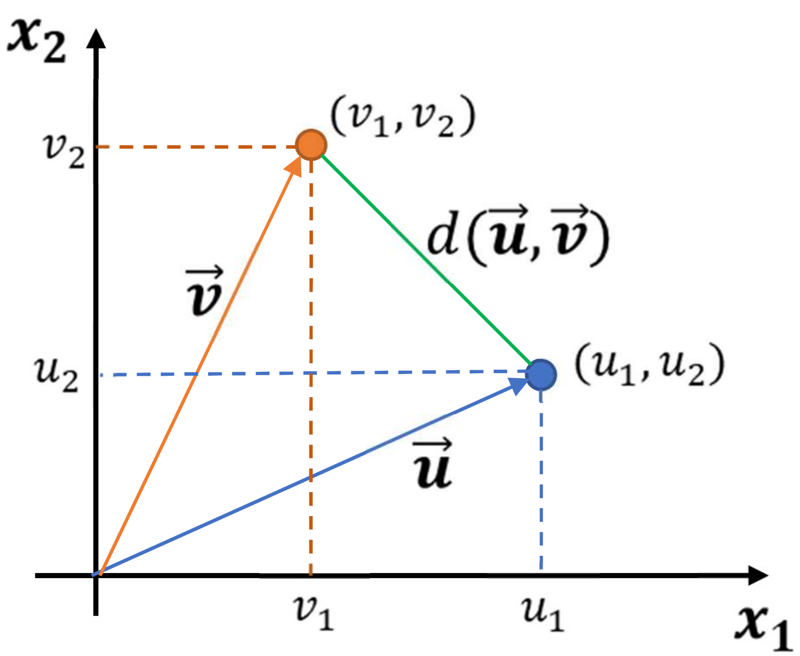
Sourav Karmakar

souravkarmakar29@gmail.com

OUTLINE

- Euclidean Distance
- Other Distance Metrics
- K-NN Classifier
- Decision Boundary of K-NN Classifier
- Choosing the value of K
- Merits and Demerits of K-NN classifier

EUCLIDEAN DISTANCE



- Consider the points in two-dimension. Each point in two-dimension can be represented by a vector of dimension two.
- The point (u_1, u_2) can be represented by the vector $\vec{\boldsymbol{u}} = [u_1, u_2]^T$
- And the point (v_1, v_2) can be represented by the vector $\vec{\boldsymbol{v}} = [v_1, v_2]^T$
- The Euclidean distance between the points is:

$$d(\vec{\boldsymbol{u}}, \vec{\boldsymbol{v}}) = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2}$$

- In general a point in n-dimensional space is represented as $\vec{\boldsymbol{u}} = [u_1, u_2, u_3, \ldots, u_n]^T$, a n-D vector
- Hence, the Euclidean distance between two points in n-dimensional space is represented as:

$$d(\vec{\boldsymbol{u}}, \vec{\boldsymbol{v}}) = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + (u_3 - v_3)^2 + \ldots + (u_n - v_n)^2} = \sqrt{\sum_{i=1}^{n} (u_i - v_i)^2}$$

- In general in vector notation, the Euclidean distance is written as: $d(\vec{\boldsymbol{u}}, \vec{\boldsymbol{v}}) = \sqrt{(\vec{\boldsymbol{u}} - \vec{\boldsymbol{v}})^T (\vec{\boldsymbol{u}} - \vec{\boldsymbol{v}})}$

OTHER DISTANCE METRICS

- **Manhattan Distance:** For two data points denoted by $\boldsymbol{x}$ and $\boldsymbol{y}$ the Manhattan distance is defined as:

$$dist_{manhattan}\,(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} |x_i - y_i|$$

- **Minkowski Distance:** For two data points denoted by $\boldsymbol{x}$ and $\boldsymbol{y}$ the Minkowski distance is defined as:

$$dist_{minkowski}\,(\boldsymbol{x}, \boldsymbol{y}, h) = \left[\sum_{i=1}^{n} (x_i - y_i)^h\right]^{\left(\frac{1}{h}\right)}$$

Note: for $h = 2$, Minkowski Distance is same as Euclidean Distance and for $h = 1$, it is Manhattan Distance

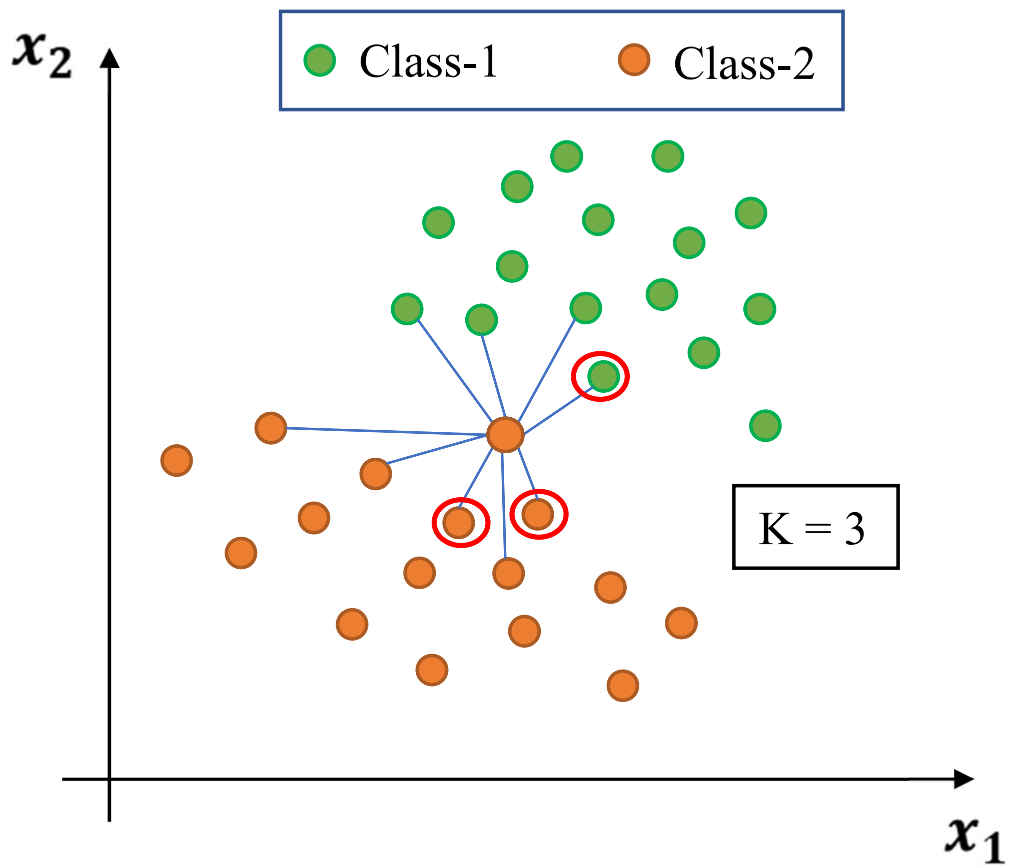
- **Chebyshev Distance:** For two data points in n-dimensional space it is defined as:

$$dist_{chebyshev}\,(\boldsymbol{x}, \boldsymbol{y}) = \max(|x_1 - y_1|,\ |x_2 - y_2|, |x_3 - y_3|, \dots, |x_n - y_n|)$$

There are other distance metric like: Mahalanobis Distance, Bhattacharya Distance etc. which are used for advanced statistical pattern recognition tasks.

K-NN CLASSIFIER

- **Intuition:** One is known by the company one keeps.



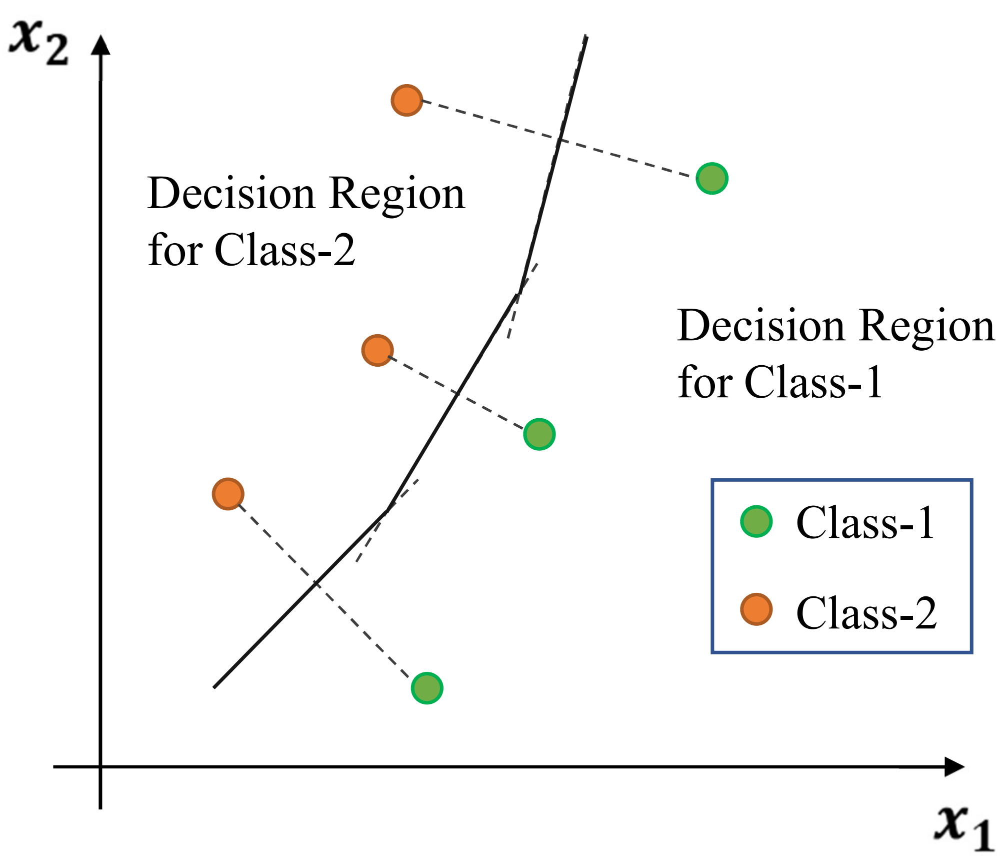
- **K-NN Algorithm:**
  1. All the training samples/ points are available beforehand.
  2. When a new test sample arrives calculate its **distance** from **all training points.**
  3. Choose K-nearest neighbours based on the distance calculated. Usually the K is a positive odd integer and supplied by user.
  4. Assign the class label of the test sample **based on majority**. i.e. for a test sample if most number of neighbours among those K-Nearest Neighbours belong to one particular class-c, then assign the class label of test sample as c.

- **Characteristics of K-NN Classifier:**

It doesn't create model based on the training patterns in advance. Rather, when a test instance comes for testing, runs the algorithm to get the class prediction of that particular testing instance. Hence, there is no learning in advance.

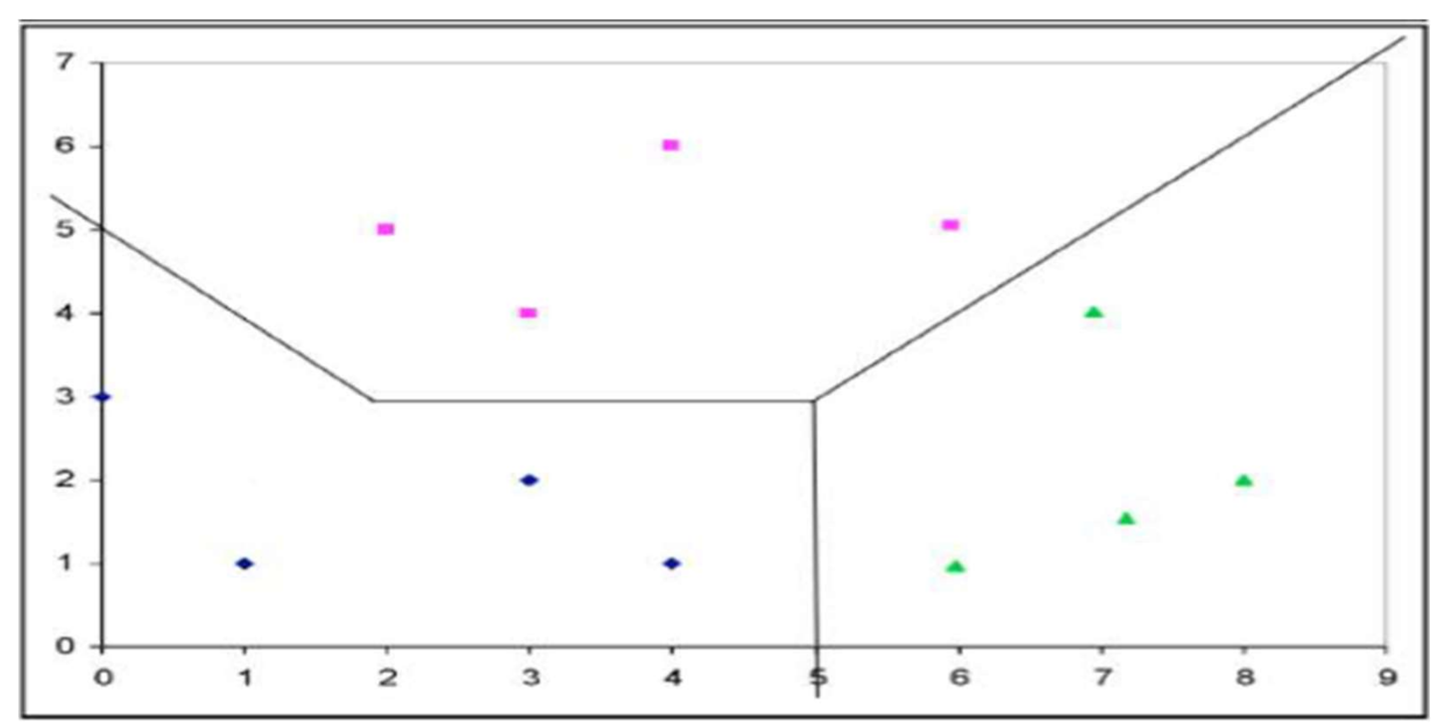
Hence, k-NN classifier is also known as **Lazy Learner**.

K-NN CLASSIFIER: DECISION BOUNDARY



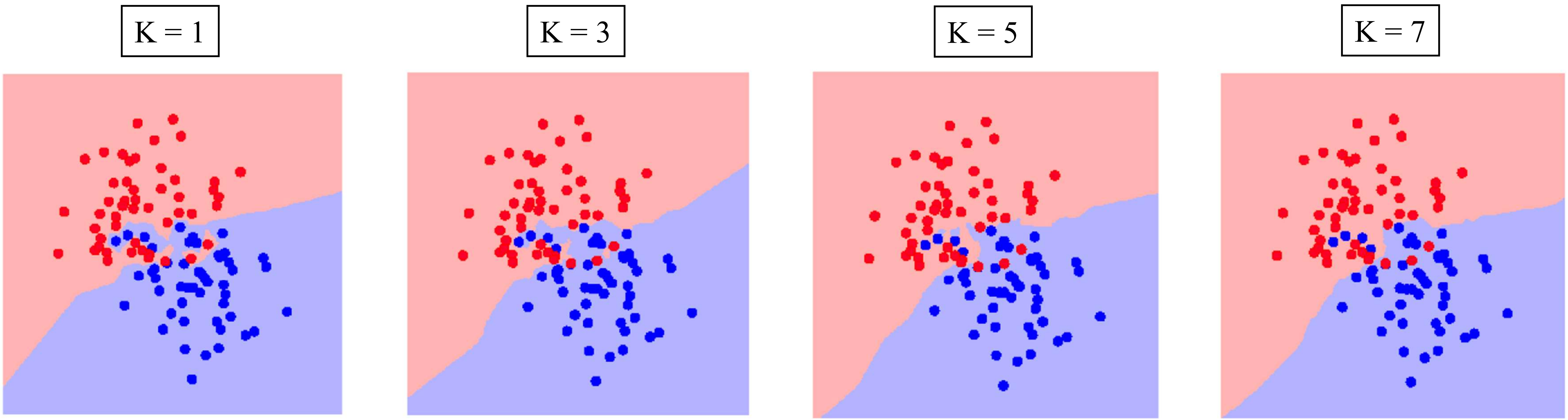
- Boundary are the points those are equidistant between the points of Class-1 and Class-2
- Construct lines between closest pairs of points in different classes.
- Draw perpendicular bisectors. End bisectors at intersections.
- Note that locally the boundary is linear.
- Hence the decision boundary is piecewise linear curve.

For multiclass classification also the same thing is done to find the decision boundary.



K-NN: CHOOSING THE VALUE OF K

- Increasing the ‘K’ simplifies the decision boundary. Because majority voting implies less emphasis on individual points



- However increasing the K also increases computational cost.

- Hence, choosing K is an optimization between how much simplified decision boundary we want vs. how much computational cost we can afford.

- Usually K = 5, 7, 9, 11 works fine for most practical problems.

K-NN CLASSIFIER: MERITS AND DEMERITS

Merits:

- K-NN Classifier often works very well for practical problems.
- It is very easy to implement, as there is no complex learning algorithm involved.
- Robust to Noisy Data.

Demerits:

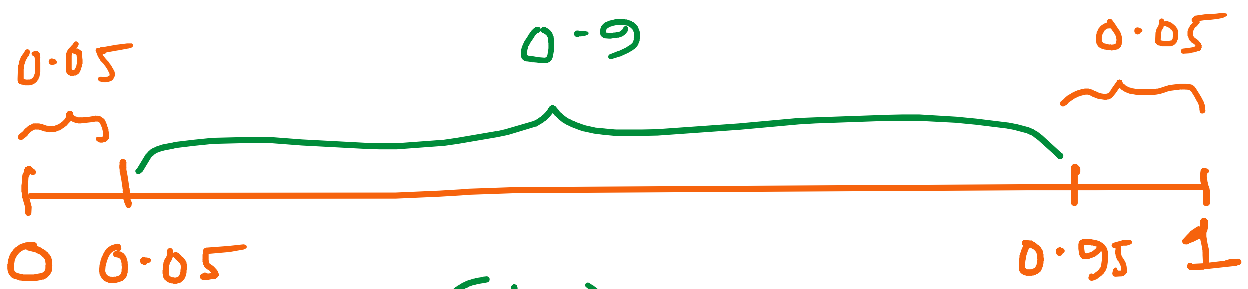
- Choosing the value of K may not be straightforward. Often the same training samples are used for different values of K, and we choose the most suitable value of K based on minimum misclassification errors on test samples.
- Doesn't work well for categorical attributes.
- Can encounter problem with sparse training data. (i.e. data points are located far away from each other)
- Can encounter problems in very high-dimensional spaces.
  - Most points are at corners.
  - Most points are at the edge of the space.

  This problem is known as ***Curse of Dimensionality*** and affect many other Machine Learning algorithms.

Curse of Dimensionality :-

→ I can randomly select any point between 0 to 1 (Uniformly distributed)

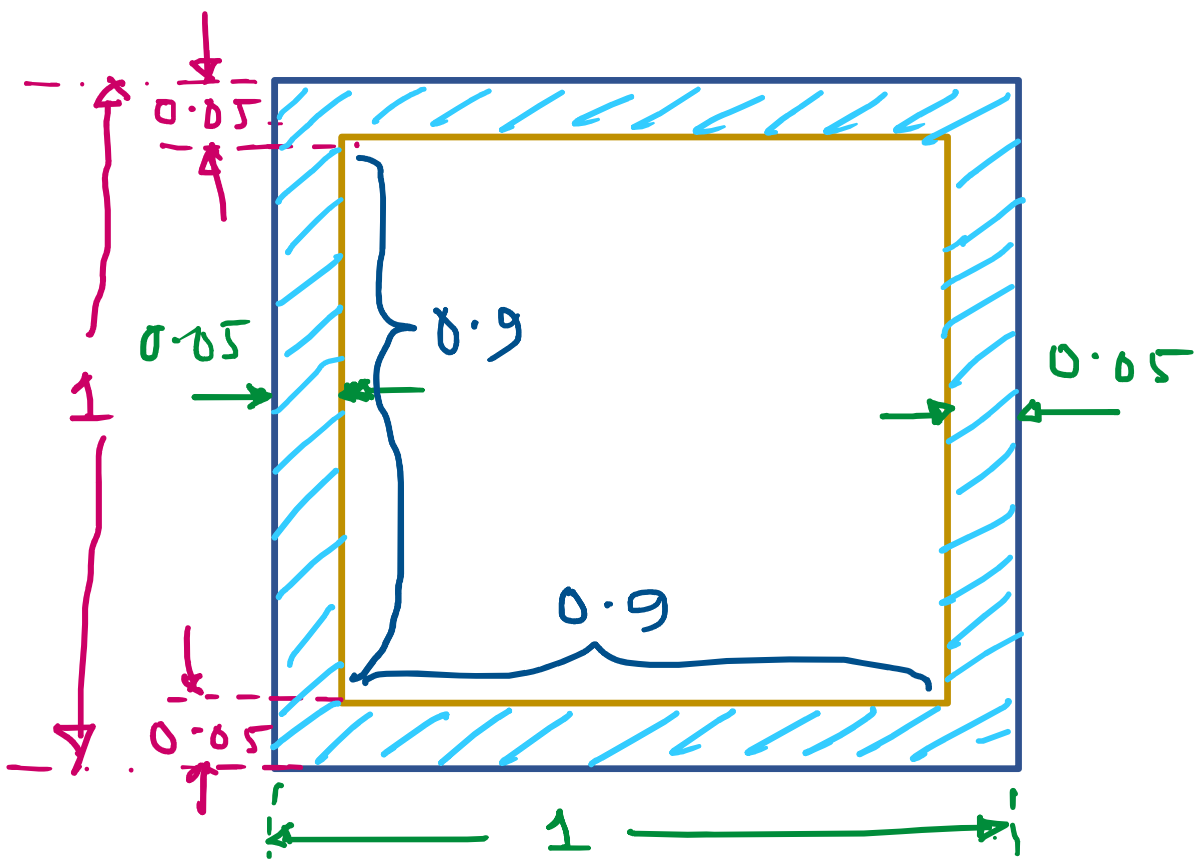
→ If the value is between $[0, 0.05)$ or $(0.95, 1)$ then we say that the value is at edge.



(1D)

(10%)

$$P(\text{selected point is at edge}) = 0.05 + 0.05 = 0.1 = 1 - (0.9)^1$$



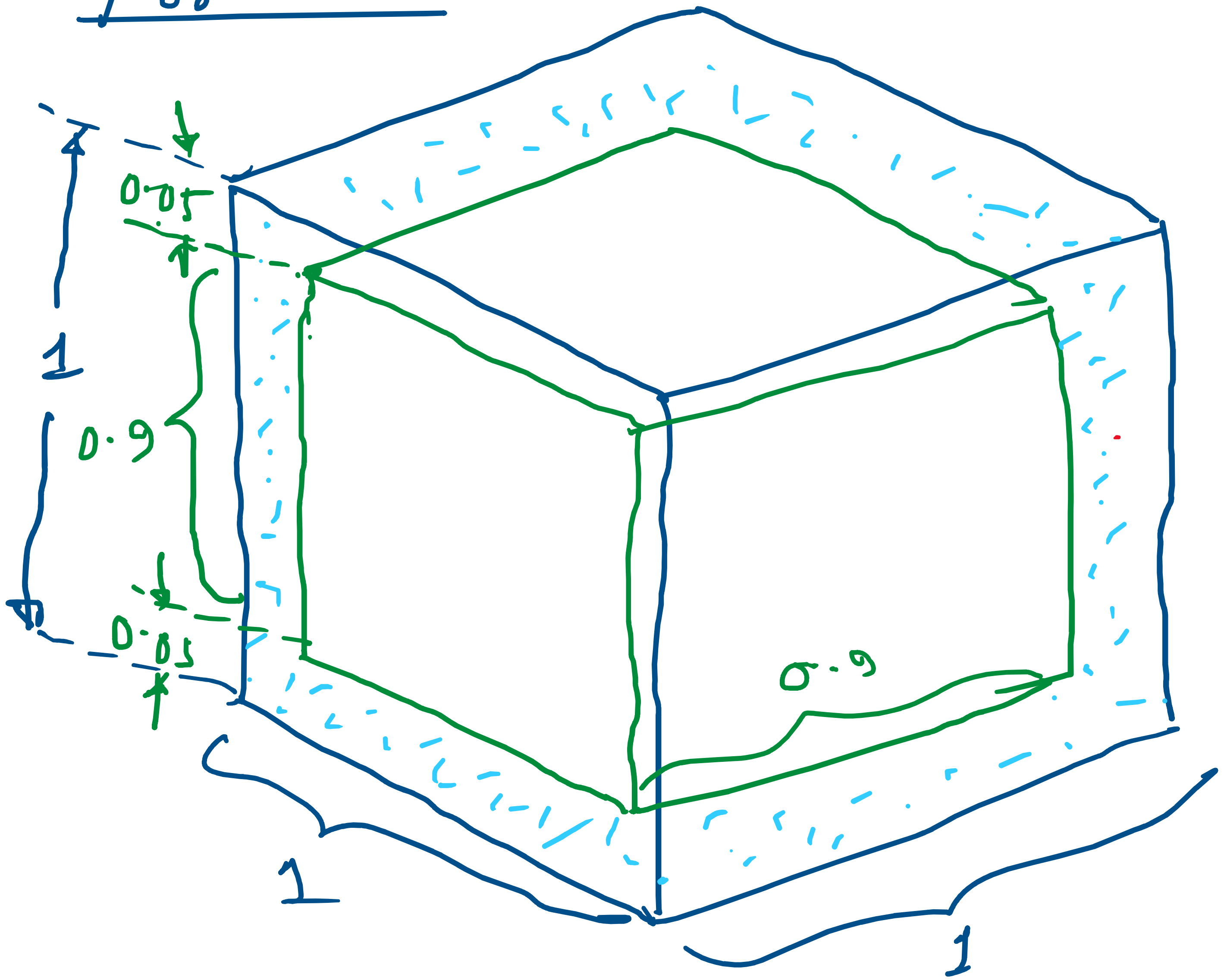
(2D)

P(selected point is on edge, i.e. in the shaded region)

$$= 1 - (0.9)^2 = 1 - 0.81 = 0.19$$

(19%)

For 3D



P(selected point is at edge)

$$= 1 - (0.9)^3$$

$$= 1 - 0.729 = 0.271 \quad (27\%)$$

Generalizing this idea for a k-dimensional hyper-cube.

P(selected point is at edge in the k dimensional cubic space)

$$= 1 - (0.9)^k$$

suppose $k = 20$, then $P = 1 - 0.1216 = 0.8784 \approx (88\%)$

Thank You