CSDS503 / COMP552 – Advanced Machine Learning

Faizad Ullah

Nearest Neighbors Methods

The K Nearest Neighbors Algorithm

- Every Machine Learning Algorithm comes up with certain assumptions.
- “A data point is known by the company it keeps” (Aesop – the data scientist)
- “A data point is known by the most common company it keeps” (Aesop – the intelligent data scientist)
- A vector is known by the company it keeps

The K Nearest Neighbors Algorithm

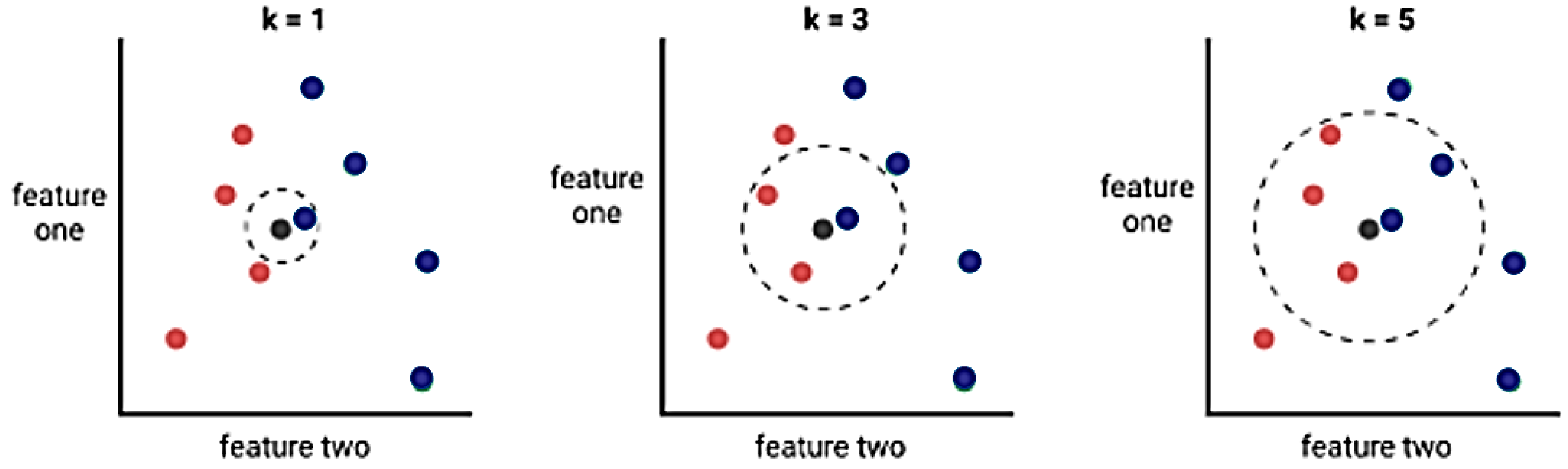
- KNN relies on the concepts of proximity and similarity.

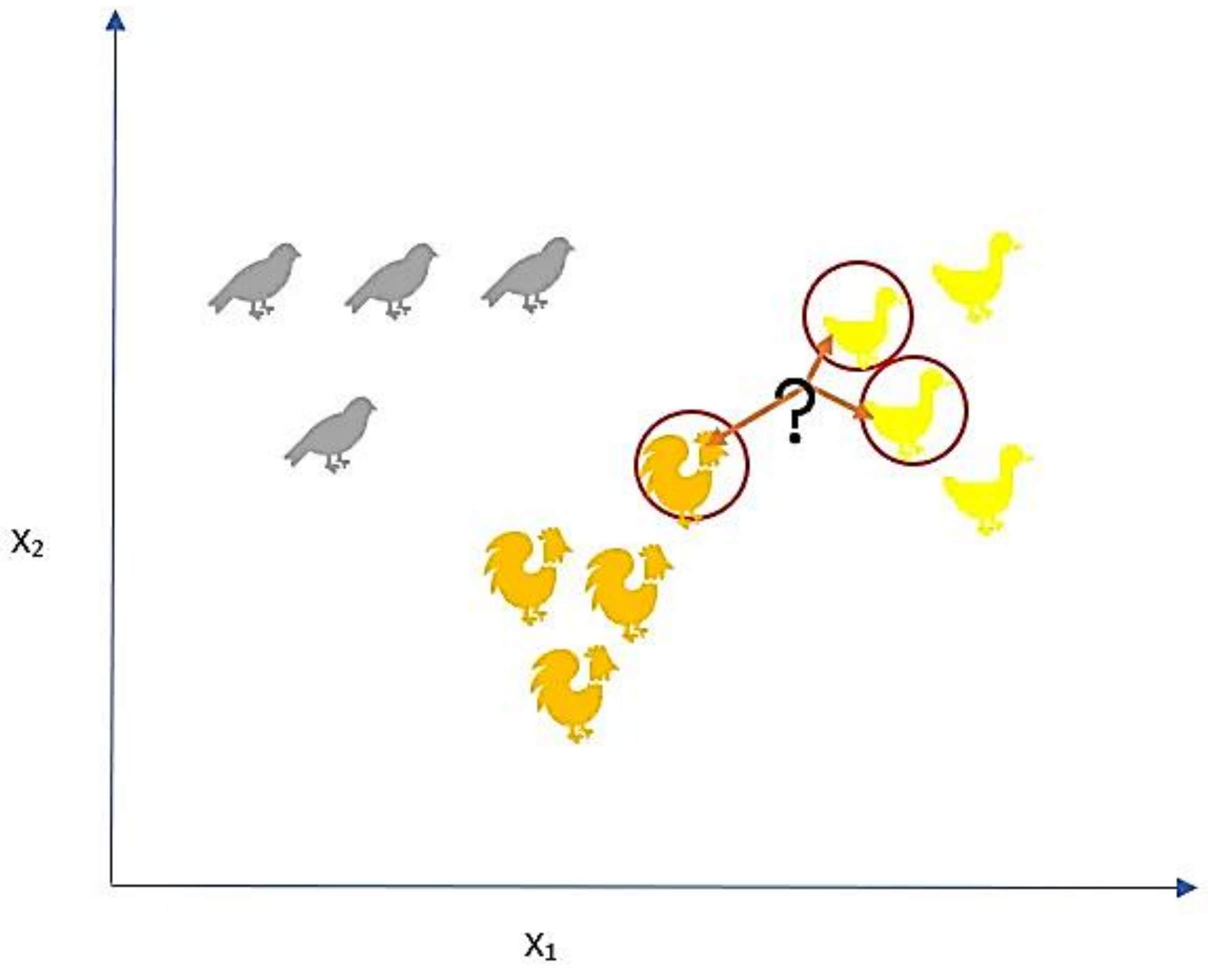
- KNN is a supervised learning algorithm capable of performing both classification and regression tasks.

The K Nearest Neighbors Algorithm

- **Basic idea:** Similar Inputs have similar outputs

- **Classification rule:** For a test input x, assign the most common label amongst its k most similar (nearest) training inputs.





https://towardsdatascience.com/an-intuitive-guide-to-knn-with-implementation-fc100bf29a6f

Formal Definition

- Assuming x to be our test point, lets denote the set of the k nearest neighbors of x as S_x
- Formally, S_x is defined as:

$$\boldsymbol{S_x \subseteq D \; s.t. \; |S_x| = k}$$
$$and$$
$$\boldsymbol{\forall (x', y') \in D \setminus S_x,}$$
$$\boldsymbol{dist(x, x') \geq \max_{(x'', y'') \in S_x} dist(x, x'')}$$

Every point that is in D but not in S_x is at least as far away from x as the furthest point in S_x

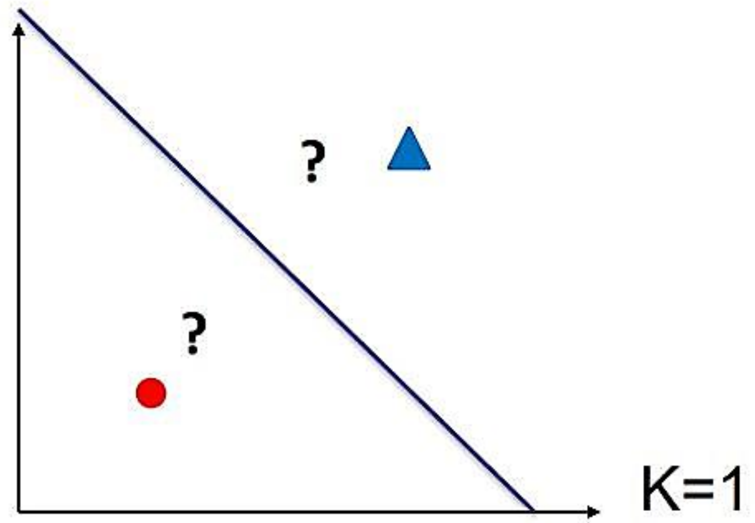
Formal Definition

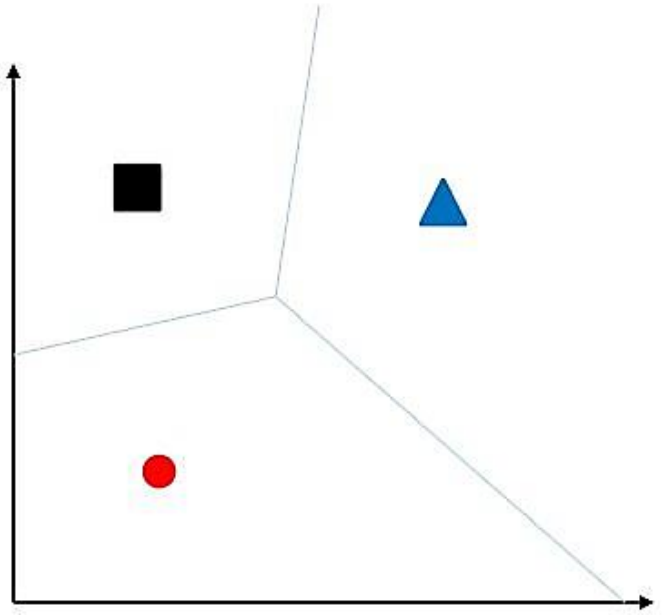
- We define the classifier $h()$ as a function returning the most common label in S_x:

$$\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{mode}(\{\boldsymbol{y}'': (\boldsymbol{x}'', \boldsymbol{y}'') \in \boldsymbol{S}_{\boldsymbol{x}}\})$$

- Where mode(·) means to select the label of the highest occurrence.
- So, what do we do if there is a draw?
- Keep k odd (for binary classification) or return the result of (k-1)NN with a smaller k.

KNN Decision Boundary





Properties of KNN – Non-parametric

The KNN Algorithm is a supervised, nonparametric algorithm.

It does not make any assumptions about the underlying distribution nor tries to estimate it.

Properties of KNN – Non-parametric

Parametric models summarize data with a fixed set of parameters (independent of the number of training examples).

No matter how much data you throw at a parametric model, it won’t change its mind about how many parameters it needs.

Properties of KNN – Non-parametric

Non-parametric models make no assumptions about the probability distribution or number of parameters when modeling the data.

Non-parametric methods are good when you have a lot of data and no prior knowledge, and when you don't want to worry too much about choosing just the right features.

Properties of KNN – Non-parametric

Non-parametric does not mean that they have no parameters!

On the contrary, **non-parametric** models (can) become more and more complex with an increasing amount of data.

Properties of KNN – Non-parametric

In a **parametric model**, we have a finite number of parameters, and in non-parametric models, the number of parameters is (potentially) infinite.

Still, the distinction is a bit ambiguous at best.

Parametric vs. Non-Parametric Models

These models have a fixed number of parameters. For example, in a linear regression model, the parameters are the slope and intercept.

These models do not have a fixed number of parameters. The number of parameters grows with the amount of training data. For example, in a k-nearest neighbors (KNN) model, the "parameters" are essentially the entire training dataset.

Parametric vs. Non-Parametric Models

Less flexible as they make strong assumptions about the data distribution

More flexible as they make fewer assumptions about the data distribution.

Parametric vs. Non-Parametric Models

Generally, less computationally intensive as they require estimating only a fixed number of parameters.

Usually, more computationally intensive as they involve a larger number of parameters and often require computation over the entire dataset.

Parametric vs. Non-Parametric Models

Higher risk of underfitting as they might not capture the underlying complexity of the data.

Higher risk of overfitting as they might capture too much noise in the data.

Parametric vs. Non-Parametric Models

1) Linear Regression

2) Logistic Regression

3) Naive Bayes

…

1) k-Nearest Neighbors

2) Decision Trees with unbounded height

3) Support Vector Machines …

Properties of KNN – Non-parametric

- Parameters and hyperparameters are two types of values that a model uses to make predictions, but they serve different purposes and are learned in different ways:
- **Parameters:**
  - These are the parts of the model that are learned from the training data during the training process.
  - For example, the weights in a linear regression model are parameters.
  - The model uses the training data to adjust the parameters to minimize the prediction error.

Properties of KNN – Non-parametric

- **Hyperparameters:**
  - These are the settings or configurations that need to be specified prior to training the model.
  - They are not learned from the data but are essential for the learning process.
  - For example, the learning rate in gradient descent, the depth of a decision tree, or the number of clusters in k-means clustering are all hyperparameters.
  - The values of hyperparameters are usually set before training the model and remain constant during the training process. They may be adjusted between runs of training to optimize model performance.

Properties of KNN – Non-parametric

- Classification: Choose the most frequent class label amongst k-nearest neighbors

- Regression: Take an average over the output values of the k-nearest neighbors and assign to the test point.

Properties of KNN – Non-parametric

- An Instance-based learning algorithm
    - Instead of performing explicit generalization, form hypotheses by comparing new problem instances with training instances
    - (+) Can easily adapt to unseen data
    - (-) Complexity of prediction is a function of n (size of training data)
- A lazy learning algorithm
    - Delay computations on training data until a query is made, as opposed to eager learning
    - (+) Good for continuously updated training data like recommender systems
    - (-) Slower to evaluate and need to store the whole training data

Distance/Similarity

Similarity/Distance Measures

- If scaled between 0 and 1, then sim = 1 − $distance$

- The Minkowski distance is a generalized metric form of Euclidean, Manhattan and Chebyshev distances

Minkowski Distance

- The Minkowski distance between two n-dimensional
- vectors P =< $p1, p2, \dots , pn$ > and Q =< $q1, q2, \dots , qn$ >, it is
- defined as:

$$d_{minkowski}(p,q) = \left(\sum_{i=1}^{n} |p_i - q_i|^a\right)^{1/a} \quad , a \geq 1$$

- a = 1, is the Manhattan distance
- a = 2, is the Euclidean distance
- $a \rightarrow \infty$, is the Chebyshev distance

Constraints on Distance Metrics

- The distance function between vectors p and q is a function $d(p, q)$ that defines the distance between both vectors is considered as a metric if it satisfy a certain number of properties:

1. Non-negativity: The distance between p and q is always a value greater than or equal to zero
   - $\boldsymbol{d(p, q) \geq 0}$
2. Identity of indiscernible vectors: The distance between p and q is equal to zero if and only if p is equal to q
   - $\boldsymbol{d(p, q) = 0 \; iff \; p = q}$

Constraints on Distance Metrics

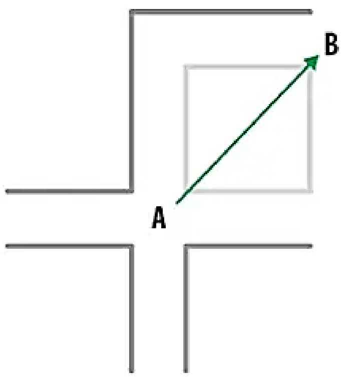
3. Symmetry: The distance between p and q is equal to the distance between q and p.

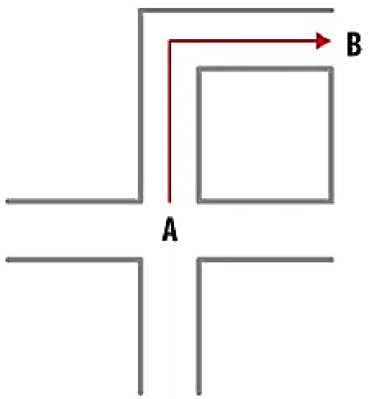
$$\boldsymbol{d}(\boldsymbol{p}, \boldsymbol{q}) = \boldsymbol{d}(\boldsymbol{q}, \boldsymbol{p})$$

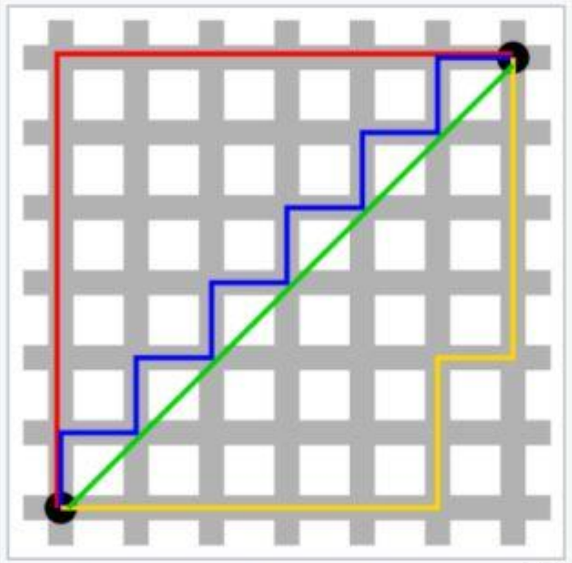
4. Triangle inequality: Given a third point r, the distance between p and q is always less than or equal to the sum of the distance between p and r and the distance between r and q

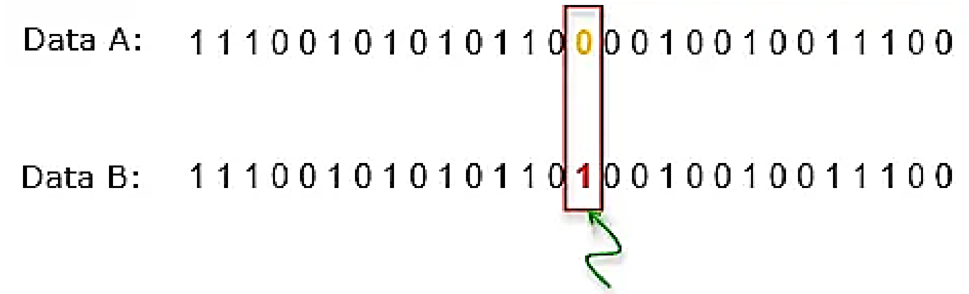
$$\boldsymbol{d}(\boldsymbol{p}, \boldsymbol{q}) \leq \boldsymbol{d}(\boldsymbol{p}, \boldsymbol{r}) + \boldsymbol{d}(\boldsymbol{r}, \boldsymbol{q})$$

Euclidean vs Manhatten vs Hamming Distance






Taxicab geometry versus Euclidean distance: In taxicab geometry, the red, yellow, and blue paths all have the same shortest path length of 12. In Euclidean geometry, the green line has length $6\sqrt{2} \approx 8.49$, and is the unique shortest path.



Data A: 1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 0 0 1 0 0 1 0 0 1 1 1 0 0

Data B: 1 1 1 0 0 1 0 1 0 1 0 1 1 0 1 0 0 1 0 0 1 0 0 1 1 1 0 0

Hamming Distance between Data A and Data B = 1

https://www.pinecone.io/learn/k-nearest-neighbor/

Manhatten Distance

- Also known as Manhattan length, rectilinear distance, L1 distance or L1 norm, city block distance, snake distance, taxi-cab metric, or city block distance

$$d_{Man}(p,q) = d(q,p) = |p_1 - q_1| + |p_2 - q_2|, \dots, |p_n - q_n|$$

$$d(p,q) = d(q,p) = \sum_{i=1}^{n} |p_i - q_i|$$

Euclidean Distance

$$d(p,q) = d(q,p) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2, \dots, (p_n - q_n)^2}$$

$$d(p,q) = d(q,p) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

Euclidean Distance

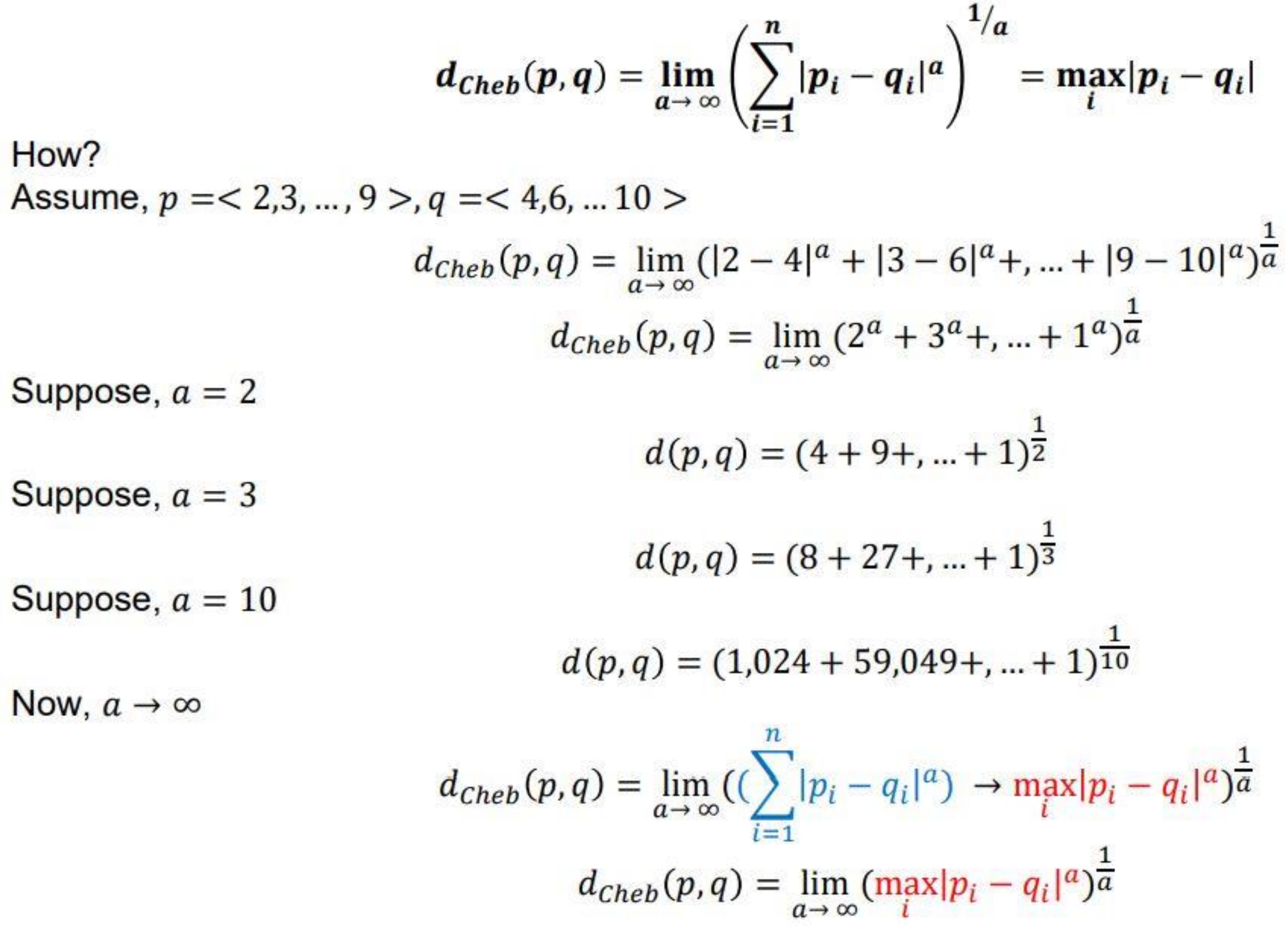
- Good choice for numeric attributes
- When data is dense or continuous, this is a good proximity measure
- Downside: Sensitive to extreme deviations in attributes (as it squares differences)
- The variables which have the largest value greatly influence the result
- Does not work well for situations where features on different scales are mixed (e.g., #bedrooms (1-5) and area (200 – 5,000 sq feet) of a house)
- Solution: feature normalization (min-max scaling)

Chebyshev Distance

- For Chebyshev distance, the distance between two vectors is the greatest of their differences along any coordinate dimension
- When two objects are to be defined as “different”, if they are different in any one dimension
- Also called chessboard distance, maximum metric, or $L\infty$ metric

$$d_{Cheb}(p,q) = \max_i |p_i - q_i|$$

Chebyshev Distance



$$\boldsymbol{d_{Cheb}(p,q) = \lim_{a\to\infty}\left(\sum_{i=1}^{n}|p_i - q_i|^a\right)^{1/a} = \max_i|p_i - q_i|}$$

How?

Assume, $p =< 2,3,\ldots,9 >, q =< 4,6,\ldots 10 >$

$$d_{Cheb}(p,q) = \lim_{a\to\infty}(|2-4|^a + |3-6|^a+,\ldots + |9-10|^a)^{\frac{1}{a}}$$

$$d_{Cheb}(p,q) = \lim_{a\to\infty}(2^a + 3^a+,\ldots + 1^a)^{\frac{1}{a}}$$

Suppose, $a = 2$

$$d(p,q) = (4 + 9+,\ldots + 1)^{\frac{1}{2}}$$

Suppose, $a = 3$

$$d(p,q) = (8 + 27+,\ldots + 1)^{\frac{1}{3}}$$

Suppose, $a = 10$

$$d(p,q) = (1{,}024 + 59{,}049+,\ldots + 1)^{\frac{1}{10}}$$

Now, $a \to \infty$

$$d_{Cheb}(p,q) = \lim_{a\to\infty}((\sum_{i=1}^{n}|p_i - q_i|^a) \to \max_i|p_i - q_i|^a)^{\frac{1}{a}}$$

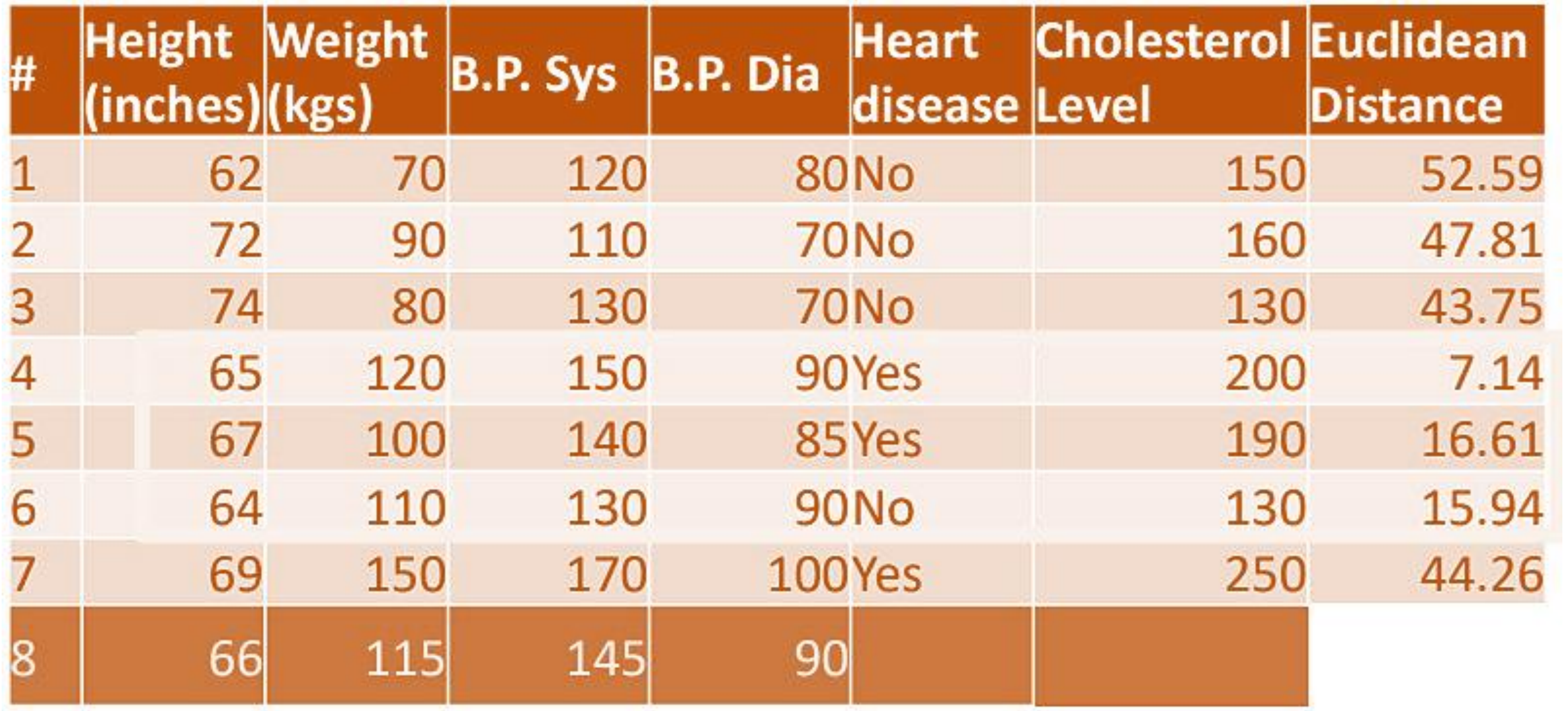
$$d_{Cheb}(p,q) = \lim_{a\to\infty}(\max_i|p_i - q_i|^a)^{\frac{1}{a}}$$

The KNN Algorithm

The KNN Algorithm

- Input:
  - Training samples D = { $(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_n, y_n)$}
  - Test sample $d = (x, y)$
  - Assume x to be an m-dimensional vector
- Output: Class label of test sample d
  1. Compute the distance between d and every sample in D
  2. Choose the K samples in D that are nearest to d; denote the set by $S_d \in D$
  3. Assign d the label y_i of the majority class in S_d
- Note:
  - All action takes place in the test phase; the training phase is essentially to clean, normalize, and store the data

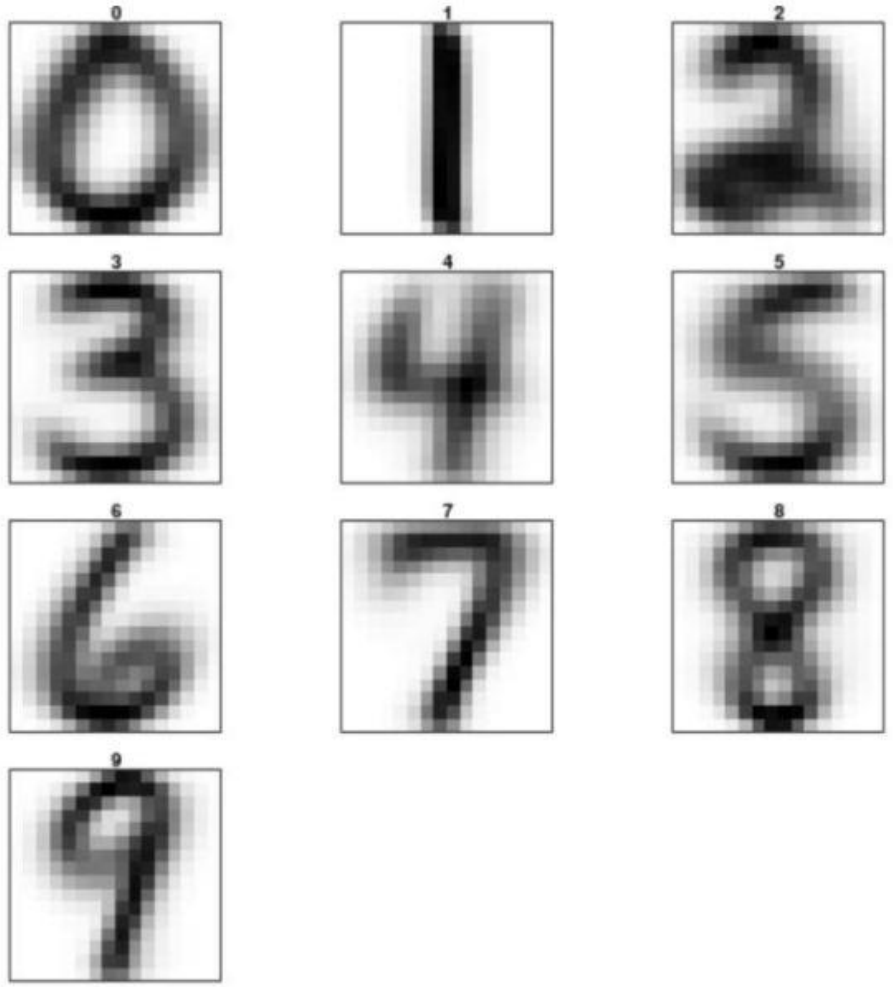
The KNN Algorithm



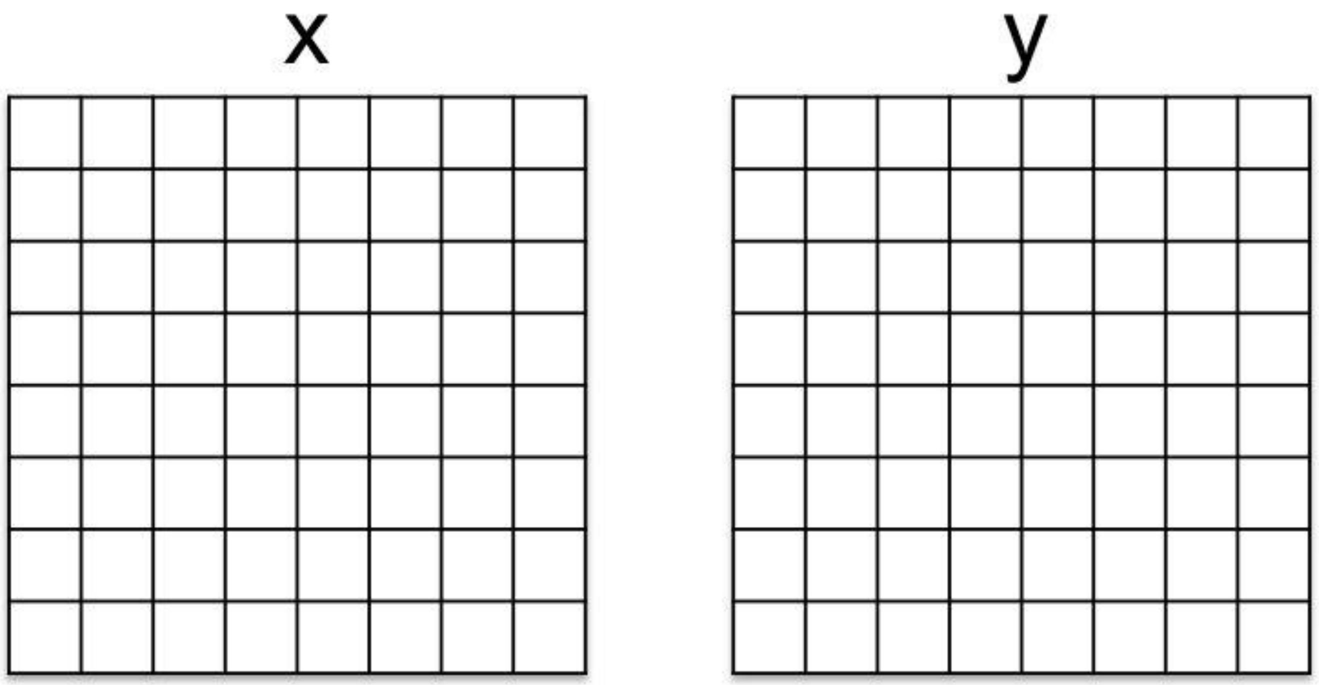
#	Height (inches)	Weight (kgs)	B.P. Sys	B.P. Dia	Heart disease	Cholesterol Level	Euclidean Distance
1	62	70	120	80	No	150	52.59
2	72	90	110	70	No	160	47.81
3	74	80	130	70	No	130	43.75
4	65	120	150	90	Yes	200	7.14
5	67	100	140	85	Yes	190	16.61
6	64	110	130	90	No	130	15.94
7	69	150	170	100	Yes	250	44.26
8	66	115	145	90			

The KNN Algorithm

- Handwritten digit recognition
  - 16x16 bitmaps
  - 8-bit grayscale
  - Euclidean distances over raw pixels



The KNN Algorithm



$$D(x,y) = \sqrt{\sum_{i=0}^{255} (x_i - y_i)^2}$$

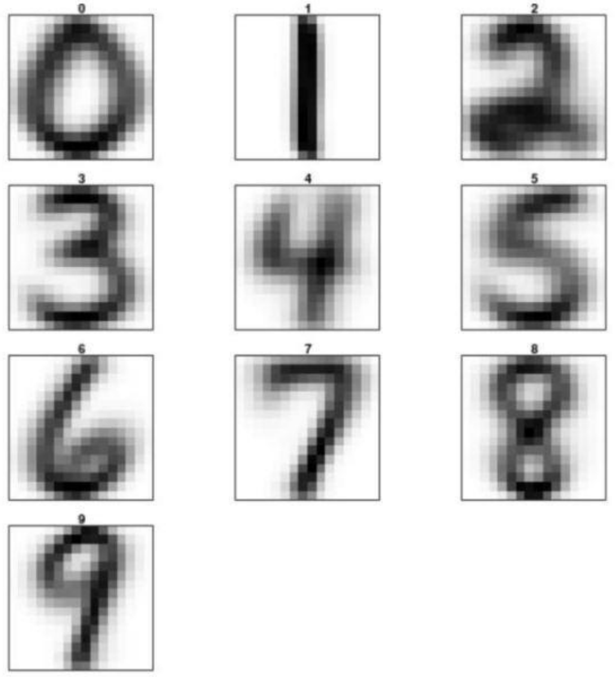
- **Accuracy:**
  - 7-NN ~ 95.2%
  - SVM ~ 95.8%
  - Humans ~ 97.5%

The KNN Algorithm

1. Text Classification using KNN:

```
1.  Machine learning is fascinating.

2.  NLP is essential.

3.  Deep learning is interesting.

4.  Text classification is essential.

5.  Data science is interesting.
```

2. Image Classification using KNN:



The KNN Algorithm

Document	machine	learning	is	fascinating	NLP	essential	deep	interesting	text	classification	data	science	Label
1	1	1	1	1	0	0	0	0	0	0	0	0	ML
2	0	0	1	0	1	1	0	0	0	0	0	0	TEXT
3	0	1	1	0	0	0	1	1	0	0	0	0	ML
4	0	0	1	0	0	1	0	0	1	1	0	0	TEXT
5	0	0	1	0	0	0	0	1	0	0	1	1	?

Time Complexity of KNN

Complexity of KNN

- Input:
  - Training samples D = { (x_1, y_1), (x_2, y_2), ..., (x_n, y_n)}
  - Test sample d = (x_1, y_1), k. Assume x to be an m-dimensional vector.
- Output: Class label of test sample d
  1. Compute the distance between d and every sample in D; n samples, each is m-dimensional $\Rightarrow O(mn)$
  2. Choose the K samples in D that are nearest to d; denote the set by $S_d \in D$
     - Either naively do K passes of all samples costing $O(n)$ each time for $O(nk)$
     - Or use the quickselect algorithm to find the kth smallest distance in $O(n)$ and then return all distances no larger than the kth smallest distance. This will accumulate to $O(n)$
  3. Assign d the label y_i of the majority class in S_d, this is $O(k)$.

Complexity of KNN

- Time complexity:

$$O(mn + n + k) = O(mn)$$

- Space complexity:

$$O(mn)$$

Tuning the Hyperparameter K

- Divide your training data into training and validation sets.
- Do multiple iterations of m-fold cross-validation, each time with a different value of k, starting from k=1
- Keep iterating until the k with the best classification accuracy (minimal loss) is found

Tuning the Hyperparameter K

- What happens if we use the training set itself as the test dataset instead of a validation set? Which k wins?
- K=1:
  - As there is always the nearest instance with the correct label: The instance itself!
- K=n:
  - KNN will always return the majority class in the dataset

Tuning the Hyperparameter K

- KNN is a simple algorithm but is highly effective for solving various real-life classification problems.
- Especially when the datasets are large and continuously growing.
- Challenges:
    - How to find the optimum value of K?
    - How to find the right distance function?
- Problems:
    1. High computational time cost for each prediction.
    2. High memory requirement as we need to keep all training samples.
    3. The curse of dimensionality.