**Machine Learning : Geometry and Nearest Neighbors**

**Nearest Neighbor Algorithm**

* Supervised Learning Technique
* Geometric Learning Algorithm
  + This algorithm uses the notion of distance and a multidimensional space between data points to compare similarity.

**How can we assign numerical values to features with multiple values?**

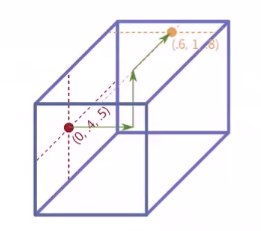
* One Hot Encoding
  + Turn a feature with k values 🡪 k features with binary values**.**
    - In other words, turn a feature with multiple values into multiple features with binary values.

**Instance as vector in high-dimensional feature space**

* If our data contains D features, then the vector representing the data point has D components.
  + **EX:**  ; is value for feature of x;
* (low dimensional problem; Think about play tennis example)
* (medium dimensional problem; Think about sensor data)
* (high dimensional problem; Think about text documents)

**K-Nearest Neighbors (KNN) Classifier**

* Inductive bias: Label for an instance should be similar to the label of nearby points.
* Label for text example x predicted to be the same as its closest neighbors.
* Computes **distance** between instances in this D dimensional space (vector space)

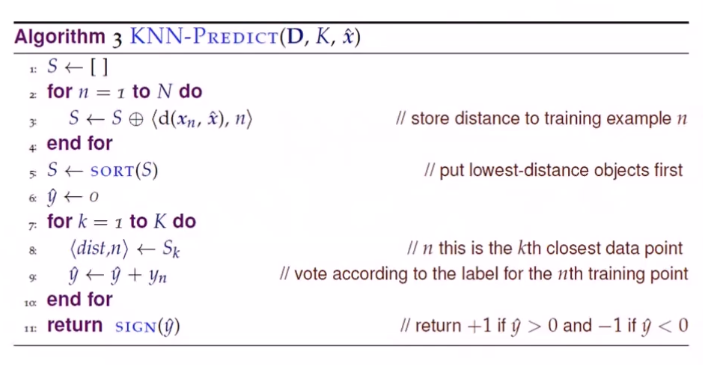


* How can we find the distance?
  + **Euclidean distance**

**EX (Euclidean Distance using the image above) :**

* + **Manhattan Distance**

**EX (Manhattan Distance using the image above):**



**D = Set of Data**

**K = Number of neighbors that you are going to consider**

**x = Particular datapoint you are trying to predict**

**S = Set of all current instances, will grow along with their distances to the new point**

**- Concatenation sign**

**n = index**

**How do we choose k?**

* Odd Number
* Test on holdout data

**Normalization**

* ; Normalizes all to fall in the range [0,1]
* def normalize(X):
* min\_value = min(X)
* range = max(X) - min\_value
* new\_list = [(x-min\_value)/range for x in X]
* return new\_list

**KNN Pros and Cons**

* **Pros**
  + Simple
  + No Training = No computational expense, in terms of runtime
  + Fairly good accuracy
  + Use for classification () or regression ()
* **Cons**
  + Expensive in terms of storage (Not ideal for big data)
    - Need to store all the data points and compute distance of all the data points.
  + Sensitive to irrelevant features, data scale, distance function