SMAI-Lecture 4

K-Nearest Neighbours Classifier

Team 3

Topics covered so far:

- *Introduction to basic concepts in ML(3-axes of ML/When to Learn/etc)*
- *ML Workflow*
- *Data Sample Representation*https://www.overleaf.com/5528281949bbhytmygtddx
- *Basic data transformations*
- *Data visualization*
- *Classification of Supervised Learning*
- *Performance Measures(Accuracy/Precision/Recall/Utility and Cost)*
- *Multi-Class problems*

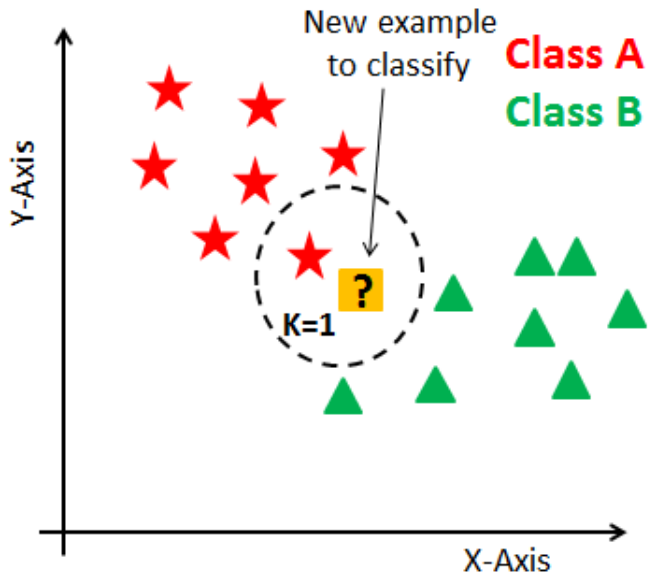
Topics covered in Lecture 4:

- Introduction to KNN
- Choosing the right value for K
- Distance Measure
- Types of Distances
- Why is knn slow?

Introduction:

KNN, Decision trees, Neural Nets are all supervised learning algorithms.Their general goal is to make accurate predictions about unknown data after being trained on known data.
Various examples were discussed during lecture in order to explain the notion of "Nearest Neighbour".Some are listed below:

- **Example 1**: As most ML algorithms work on "Assumption".In KNN the rough assumption is that **"nearby points share similar labels"**.Following example illustrates the "assumption making" step while solving a problem by intuition.



Given two classes of points in plane(class A and B). Try to classify a new element indicated by '?' among these two.

As class A is nearer to the element to classify hence intuition says that the class of unknown element should be A(which in turn is the logic behind KNN).

So,basically KNN for now can be written as:

  - Given an unknown, pick the k(say 5) closest neighbours by some distance function(This step might get messy in case of larger data set).
  - Class of unknown is the **"mode"** of the k-nearest neighbour's labels.

- **Example 2**: To illustrate the "Voronoi Diagram" and notion of "Decision Boundaries".
[Example from book by Patrick Winston] Given an assembly line of metal pieces.Two have holes and two don't.Two are larger while two are not.Try to plot them on graph

with hole area and total area being on x and y-axis respectively.
Decision making : After taking the picture of the metal piece our system should be able to place it correctly on the area graph.
Some error might get included because of the approximation in calculating the hole and total areas and hence introducing decision boundaries would help.

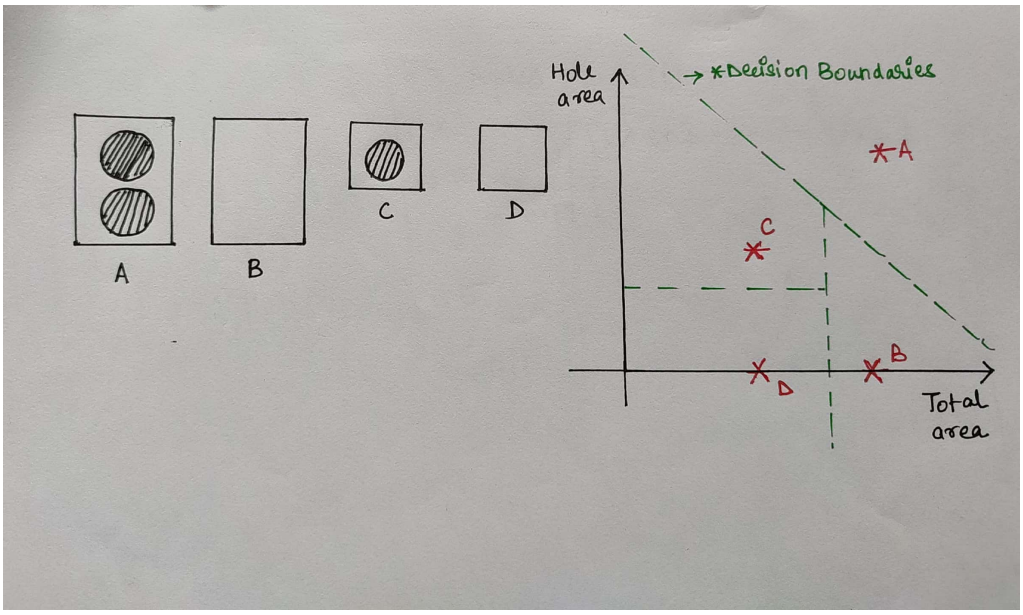


Fig 1

Decision Boundary:[wikipedia] *In a statistical-classification problem with two classes, a decision boundary or decision surface is a hypersurface that partitions the underlying vector space into two sets, one for each class.*
The classifier will classify all the points on one side of the decision boundary as belonging to one class and all those on the other side as belonging to the other class.

Voronoi Diagram:A Voronoi diagram of a set of "sites" (points) is a collection of regions that divide up the plane. Each region corresponds to one of the sites, and all the points in one region are closer to the corresponding site than to any other site.

Class doubts bullets:

- Decision Boundaries need not be linear everytime.
- What about on boundary data?
  Need to break the tie somehow just like while doing k nearest neighbours if there are equal neighbours from two or more different classes how would you decide which class to keep the new element in?(One possible tie breaker could be to decide as per the 1st nearest neighbour).

- **Example 3**: Intuition Document Classification
  Decision making for "which direction one might need to go in?".(For example, given a set of articles find the most appropriate ones according to the topic)

– The 'figure' vs 'laboratory' example:

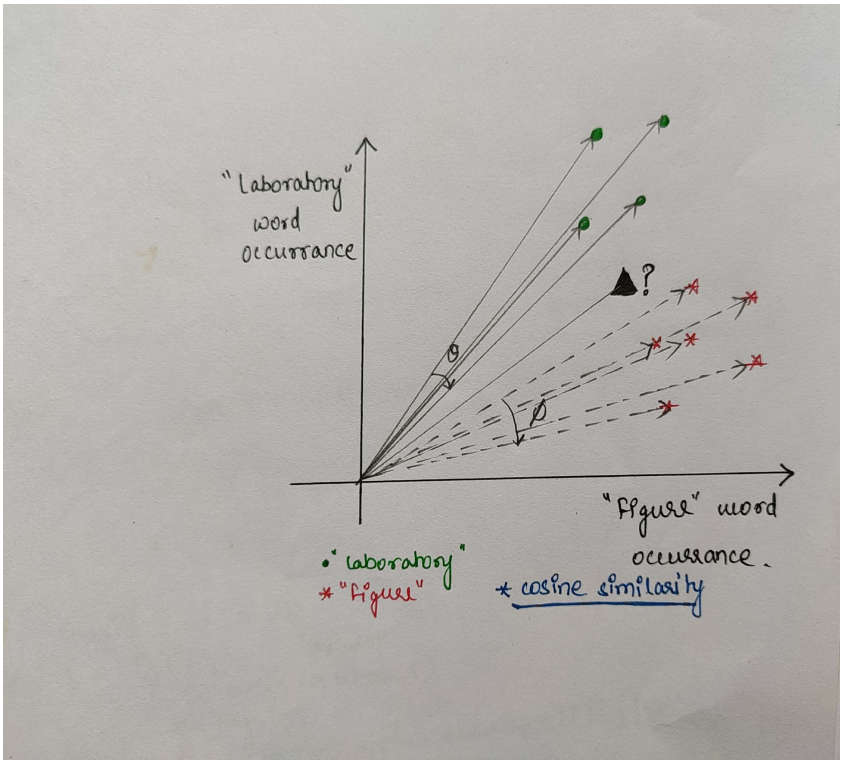
– 

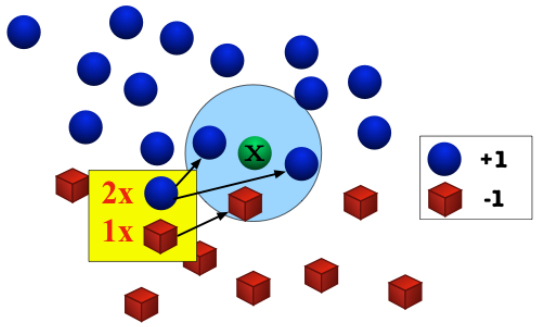
Fig 2

Euclidean distance is not appropriate to judge class here.(Which magazine would be appropriate to choose for word 'computer'? 'Wired' or 'fashion'?(Wired of course but how to reach the answer?))
Calculate cosine similarity. Notion of distance:We need to calculate distance differently in different situations.For eg:To find distance between the words "CAT" and "MAT",change in letters might be considered $(1 + 0 + 0 = 1)$.

Linear Classification: Discussed in detail in Lecture-6.
What should be the appropriate value of k??

KNN Algorithm

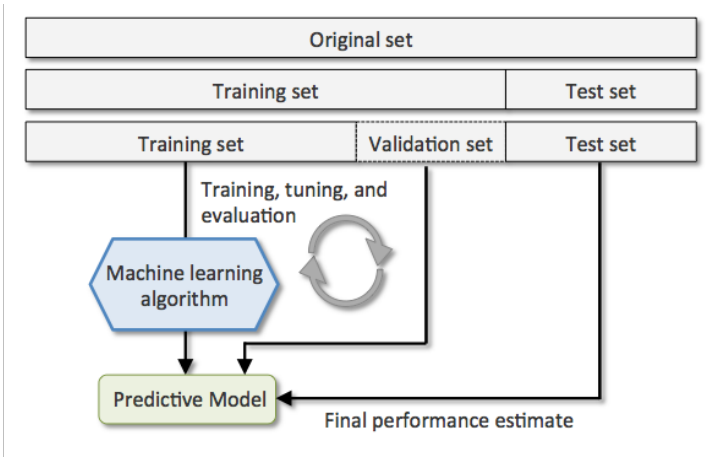
Assumption: Similar Inputs have similar outputs.
Classification rule: For a test input x, assign the most common label amongst its k most similar training inputs.
A binary classification example with k=3. The green point in the center is the test sample x. The labels of the 3 neighbors are 2(+1) and 1(-1) resulting in majority predicting (+1).



Given: Pairs of training examples x_i, y_i , 1 <= i <= m, where m is the number of training examples.
Find: Classify a testing point x_j.
Algorithm:
1. Initialize K to your chosen number of neighbours.
2. For each example in the data x_i , 1 <= i <= m,
   2.1 Calculate the distance between x_j and the current example from the data x_i.
   2.2 Add distance,y_i to an ordered collection.
3. Sort the ordered collection of distances and labels by distances in increasing order.
4. Pick the first K entries from the sorted collection.
5. Get the labels of the selected K entries.
6. If regression, return weighted mean of the K labels.
7. If classification, return mode(maximum occurrence) of K labels.

Choosing the right value for K

To select the K that's right for your data, we run the KNN algorithm several times with different values of K and choose the K by keeping a check on accuracy of predicted values on validation data. Final performance is tested on testing data.
Some points to be taken care of:
1. As we decrease the value of K to 1, our predictions become less stable.With K=1 and suppose we have a query point surrounded by several reds and one green, but green is the single nearest neighbor. Reasonably, we would think the query point is most likely red, but because K=1, KNN incorrectly predicts that the query point is green.
2. Inversely, as we increase the value of K, our predictions become more stable due to



majority voting/averaging, and thus, more likely to make more accurate predictions (up to a certain point). Eventually, we begin to witness an increasing number of errors as most common class will be the one always predicted. It is at this point we know we have pushed the value of K too far.
3. In cases where we are taking a majority vote (e.g. picking the mode in a classification problem) among labels, we usually make K an odd number to have a tiebreaker.
Rule for choosing right value of k: k $< \sqrt{n}$, where n is the number of training examples.

Distance Measure

Performance of the algorithm depends on distance measure.
Consider Euclidean Distance:
- symmetric,spherical,test all dimensions equally.
- sensitive to large difference in single attribute.
Better version would be:
D(X,X') = $(X - X')^T \sum^{-1} (X - X')$
D(X,Y) = $(\sum_{i=0}^{m} |x_i - y_i|^p)^{\frac{1}{p}}$
p=1 Manhattan Distance
p=2 Euclidean Distance
p=∞ Chebyshev Distance (max$|x_i - y_i|$)

Types of Distances

When you read the word "Distance", the basic definition of the word comes out in your mind which states it is a numerical measurement of how far apart objects or points are. When we say distance, what we mean is the distance metrics. And the basic Mathematics Definition, Distance metric uses distance function which provides a relationship metric between each element in the data set.Several Machine Learning Algorithms — Supervised or Unsupervised, use Distance Metrics to know the input data pattern to make any Data-Based decision. A good distance metric helps in improving the performance of Classification, Clustering, and Information Retrieval process significantly.

Types of distances matrices

1. Minkowski Distance
2. Manhattan Distance
3. Euclidean Distance
4. Hamming Distance
5. Cosine Similarity and Cosine Distance

Minkowski Distance

When we think about distance, we usually imagine distances between cities. That is the most intuitive understanding of the distance concept. Fortunately, this example is perfect for explaining the constraints of Minkowski distances. We can calculate Minkowski distance only in a normed vector space. Let's start by proving that a map is a vector space. If we take a map, we see that distances between cities are normed vector space because we can draw a vector that connects two cities on the map. We can combine multiple vectors to create a route that connects more than two cities. Now, the adjective "normed." It means that the vector has its length and no vector has a negative length. That constraint is met too because if we draw a line between cities on the map, we can measure its length. Again, a normed vector space is a vector space on which a norm is defined. Suppose X is a vector space then a norm on X is a real-valued function $\|x\|$ which satisfies below conditions -

1. Zero Vector- Zero vector will have zero length. To say, If we look at a map, it is obvious. The distance from a city to the same city is zero because we don't need to travel at all. The distance from a city to any other city is positive because we can't travel -20 km.
2. Scalar Factor- The direction of the vector doesn't change when you multiply it with a positive number though its length will be changed. Example: We traveled 50 km North. If we travel 50 km more in the same direction, we will end up 100 km North. The direction does not change.

3. Triangle Inequality- If the distance is a norm then the calculated distance between two points will always be a straight line.

Normed vector has the above properties which help to keep the norm induced metric- homogeneous and translation invariant. The distance can be calculated using the below formula -

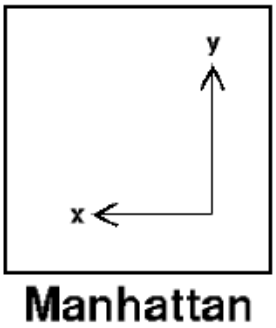
$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

Minkowski distance is the generalized distance metric. Here generalized means that we can manipulate the above formula to calculate the distance between two data points in different ways. As mentioned above, we can manipulate the value of p and calculate the distance in three different ways-

1. p = 1, Manhattan Distance
2. p = 2, Euclidean Distance
3. p = ∞, Chebychev Distance

Manhattan Distance

We use Manhattan Distance if we need to calculate the distance between two data points in a grid-like path. As mentioned above, we use the Minkowski distance formula to find Manhattan distance by setting p's value as 1. Let's say, we want to calculate the distance, d, between two data points- x and y.



Distance d will be calculated using an absolute sum of the difference between its cartesian coordinates as below :

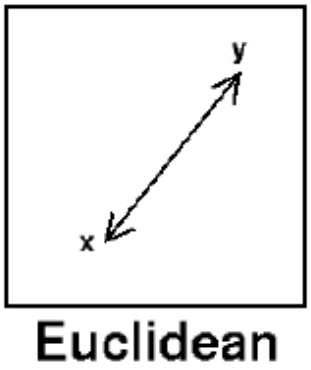
$$d = \sum_{i=1}^{n} |x_i - y_i|$$

Euclidean Distance

Euclidean distance is one of the most used distance metrics. It is calculated using the Minkowski Distance formula by setting p's value to 2. This will update the distance d' formula as below

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Euclidean distance formula can be used to calculate the distance between two data points in a plane.



Hamming Distance

A Hamming distance in information technology represents the number of points at which two corresponding pieces of data can be different. It is often used in various kinds of error correction or evaluation of contrasting strings or pieces of data.The Hamming distance involves counting up which set of corresponding digits or places are different, and which are the same. For example, take the text string "hello world" and contrast it with another text string, "herra poald." There are five places along the corresponding strings where the letters are different.
Example: Suppose there are two strings 1101 1001 and 1001 1101.
11011001 $\oplus$ 10011101 = 01000100. Since, this contains two 1s, the Hamming distance, d(11011001, 10011101) = 2.

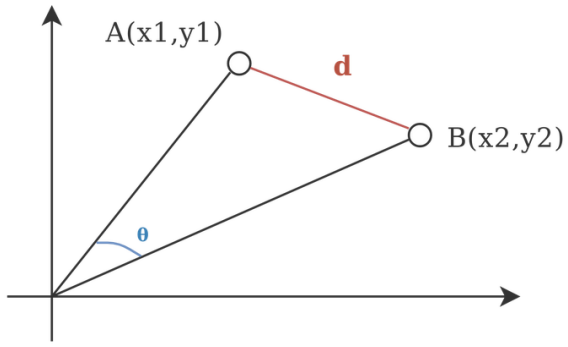
Cosine Similarity and Cosine Distance

There are two terms: Similarity and Distance. They are inversely proportional to each other i.e if one increases the other one decreases and vice-versa. the formula for which is:
1 - cos_sin = cos_distance
Cosine similarity formula can be derived from the equation of dot products:-

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos\theta$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

Distance d will be calculated using an absolute sum of the difference between its cartesian coordinates as below :



Now, you must be thinking which value of cosine angle will help find out the similarities.

Cos 0′ = 1 Cos 90′ = 0
Cos 180′ = -1

Now that we have the values which will be considered to measure the similarities, we need to know what do 1, 0 and -1 signify.
Here cosine value 1 is for vectors pointing in the same direction i.e. there are similarities between the documents/data points. At zero for orthogonal vectors i.e. Unrelated(some similarity found). Value -1 for vectors pointing in opposite directions(No similarity).
To find the cosine distance, we simply have to put the values in the formula and compute.

kNN Practical issues

Resolving ties:

1. Take equal number of positive/negative neighbours.
2. Use of odd value of k is preferable(does not solve multi class).To elude ties, odd numbered k's are preferred. However, tie scores in k-NN indicate low confidence in the prediction.

Breaking ties:

1. Random: Flip a coin to decide positive / negative, i.e. randomly assign a class.
2. Pick class with greater prior, i.e. the class that is observed most in the dataset.
3. Use 1-NN classifier to decide.

Missing values:

1. One has to "fill in", otherwise can't compute distance.
2. If the value of a given attribute A is missing in tuple X1 and/or in tuple X2, we assume the maximum possible difference.

3. For categorical attributes, we take the difference value to be 1 if either one or both of the corresponding values of A are missing.
4. If A is numeric and missing from both tuples X1 and X2, then the difference is also taken to be 1.

How to determine a good value for k?

- Starting with k = 1, we use a test set to estimate the error rate of the classifier.
- The k value that gives the minimum error rate may be selected.

Complexity

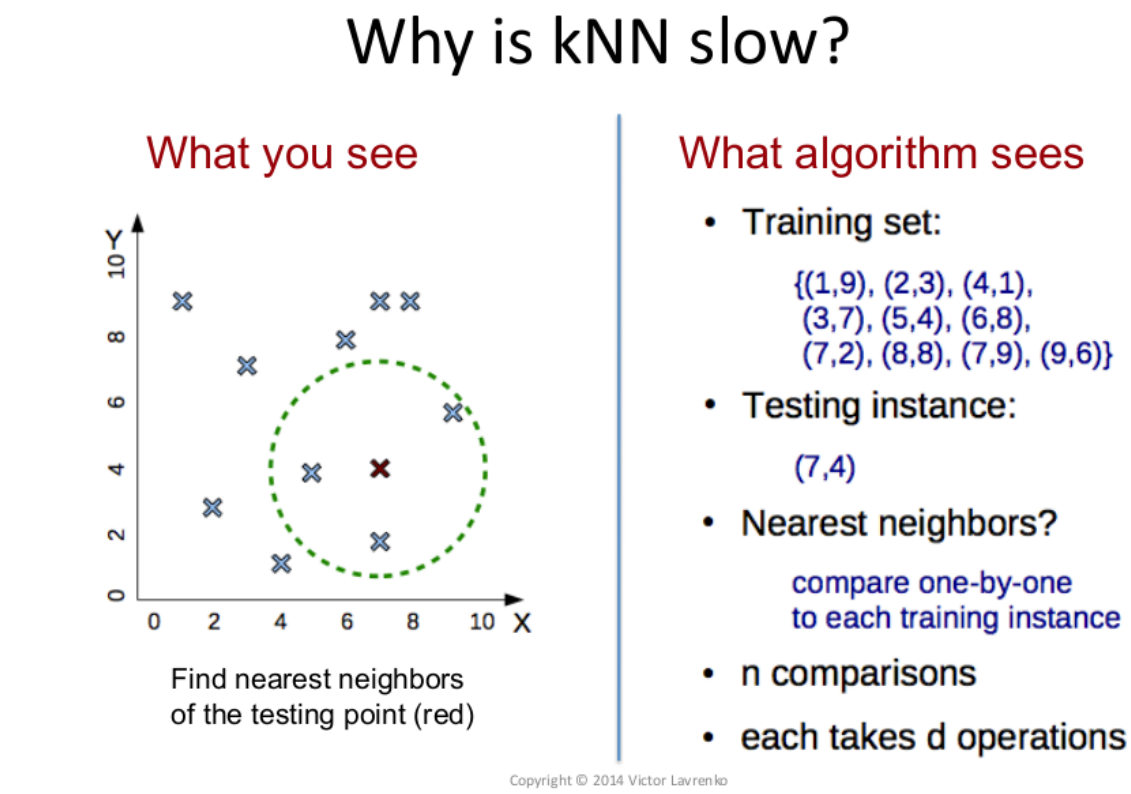
- Basic kNN algorithm stores all examples.
- Suppose we have n examples each of dimensions d.
- It takes O(d) to compute distance to one example.
- It takes O(nd) to computed distances to all examples.
- It takes additional O(nk) time to find k closest examples.
- Thus the total time is O(nk+nd).
- kNN becomes very expensive for a large number of samples.

Advantages of KNN classifier

- Can be applied to the data from any distribution for example, data does not have to be separable with a linear boundary
- It is non parametric.
- It has a zero cost of the learning process.
- Sometimes it is robust with regard to noisy training data.
- Well suited for multimodal classes.(Multimodality refers to the use of more than one mode, i.e. linguistic, textual, spatial, and visual modes, etc. in communication practices to create a particular message or set of messages).
- It is easier to update in online setting.
- The algorithm is simple and easy to implement.
- There's no need to build a model, tune several parameters, or make additional assumptions.
- The algorithm is versatile. It can be used for classification, regression, and search.
- It gives good classification if the number of samples is large enough.

Disadvantages of KNN classifier

- Choosing k may be tricky.
- The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase. Computationally expensive and high test time.
- There's a need to handle missing values.
- It is very sensitive to the scale of data as well as irrelevant features.
- It has no training stage, all the work is done during the test stage.
- This is actually the opposite of what we want. Usually we can afford training step to take a long time, but we want fast test step.
- Performance of the algorithm depends on the dimensions used.



Applications of kNN Classifier

1. Used in classification.
2. Used to get missing values.

3. Used in pattern recognition.
4. Used in gene expression.
5. Used in protein-protein prediction.
6. Used to get 3D structure of protein.
7. Used to measure document similarity.

Making kNN faster

Training takes O(d), but testing takes O(nd) time complexity.
Reduce d:Dimensionality reduction
simple feature selection, other methods O(d^3)

Reduce n: don't compare to all training*examples

Quickly identify m¡¡n potential neighbours and compare to them. That is pick k nearest neighbours O(md) time.

K-D trees: low dimensional, real valued data O(d log 2 n), only works when d¡¡n, in exact.

Inverted list, high dimensional discrete data.

Locality sensitive hashing, high-d, discrete or real valued.