CS1138

Machine Learning

Lecture : k - Nearest Neighbors

(Slide Credits: Sebastian Raschka)

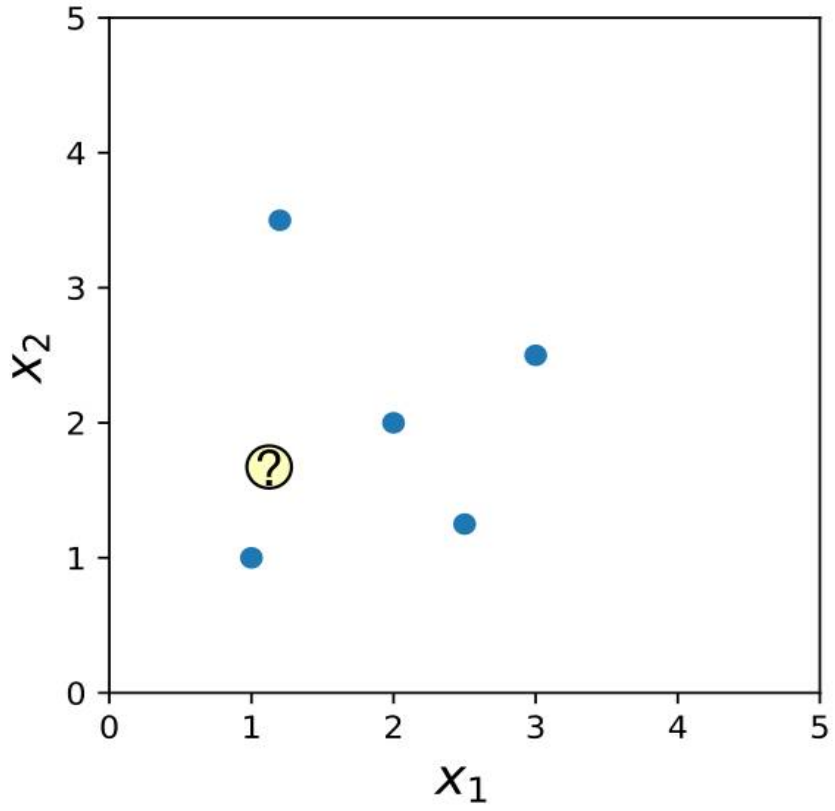
Arpan Gupta

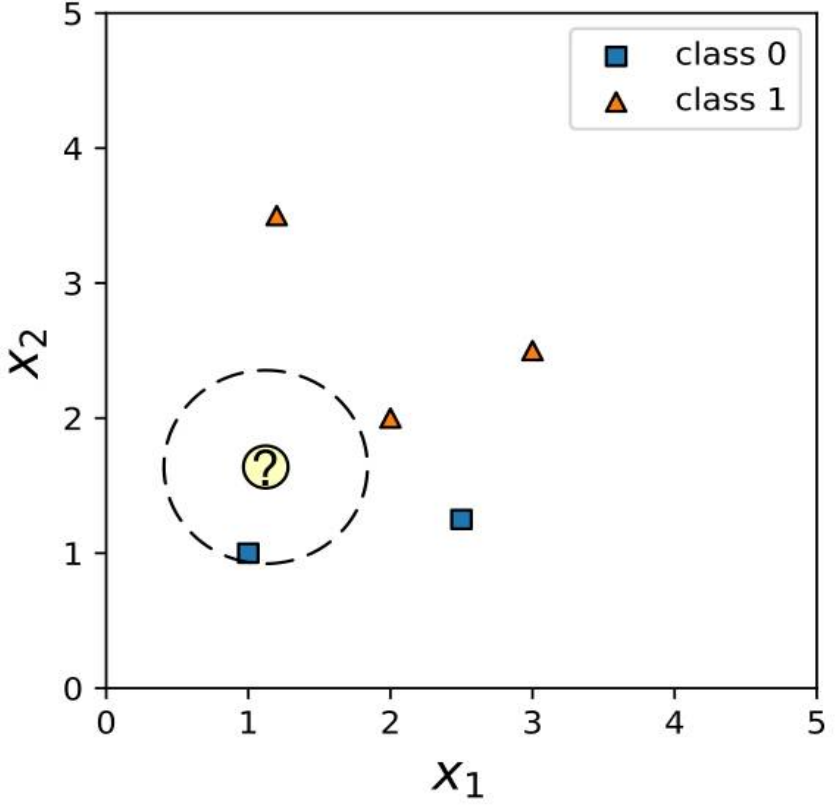
Overview

- 1 – Nearest Neighbor
- Nearest Neighbor Decision Boundary
- k – Nearest Neighbors
- Distance Measures
- Curse of Dimensionality

1 – Nearest Neighbour

Task: predict the target / label of a new data point





How? Look at most "**similar**" data point in training set.
In this example, it can be “predicted” as a square since the closest point is a square.

Nearest Neighbours

- It makes the assumption, that data points that are “close” together, have the same labels.
  - We may think of datasets where this does not hold, but k-NN makes this assumption.

Nearest Neighbour Classifier: Algorithm

- Consider a new point arrives, which is to be assigned a class label.
- Check the distances of the new point with the existing data points.
- Take k “closest” points to this new point and let them vote for this new point
- The label that gets the max vote is assigned as the label to this new point.

1 – Nearest Neighbour: Training Step

- Nearest Neighbour classifiers are also called as “Lazy Learners”
- $(x^{(i)}, y^{(i)})$ is the training set with N samples

$$\langle \mathbf{x}^{[i]}, y^{[i]} \rangle \in \mathscr{D} \quad (|\mathscr{D}| = n)$$

How do we "train" the 1-NN model?

1 – Nearest Neighbour: Training Step

$$\langle \mathbf{x}^{[i]}, y^{[i]} \rangle \in \mathscr{D} \quad (|\mathscr{D}| = n)$$

To train the 1-NN model, we simply "remember" the training dataset

1 – Nearest Neighbour: Prediction Step

Given: $\langle \mathbf{x}^{[i]}, y^{[i]} \rangle \in \mathcal{D} \quad (|\mathcal{D}| = n)$

$\langle \mathbf{x}^{[q]}, ??? \rangle$

To predict the label of query point

Predict: $f(\mathbf{x}^{[q]})$

Algorithm:

closest_point := None

closest_distance := ∞

query point

- for $i = 1, \ldots, n$:
  - current_distance := $d(\mathbf{x}^{[i]}, \mathbf{x}^{[q]})$
  - if current_distance < closest_distance:
    - closest_distance := current_distance
    - closest_point := $\mathbf{x}^{[i]}$
- return f(closest_point)

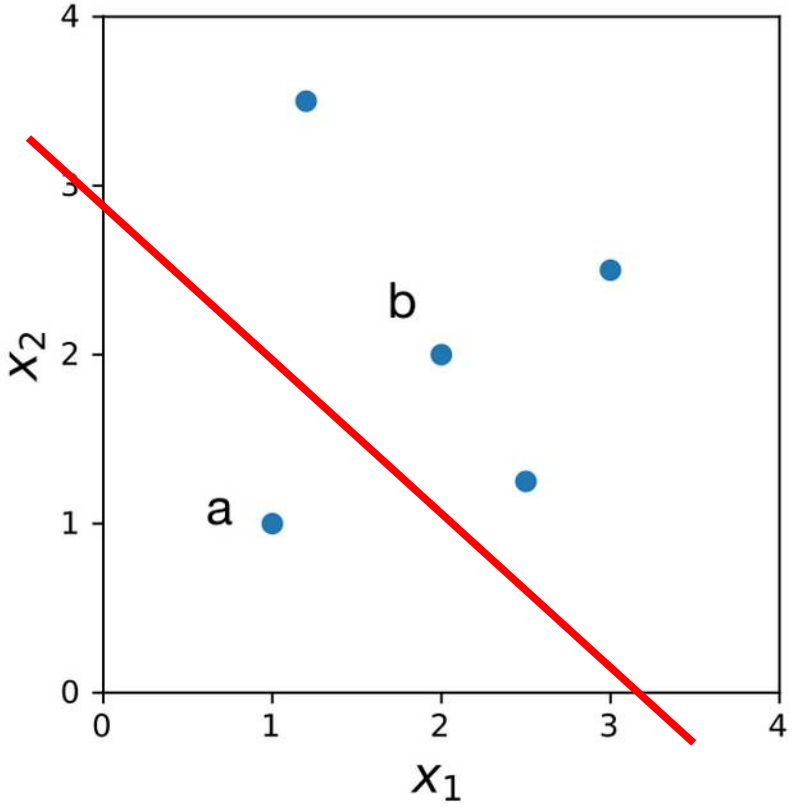
Commonly Used: **Euclidean Distance (L_2)**

$$d(\mathbf{x}^{[a]}, \mathbf{x}^{[b]}) = \sqrt{\sum_{j=1}^{m} \left(x_j^{[a]} - x_j^{[b]} \right)^2}$$

$\mathbf{x}^{[a]}$ and $\mathbf{x}^{[b]}$ are ***m*** dimensional.

It is better that the features are on a similar scale when using euclidean distance, else one feature will dominate the distance computation.

Decision Boundary between points (a) and (b)

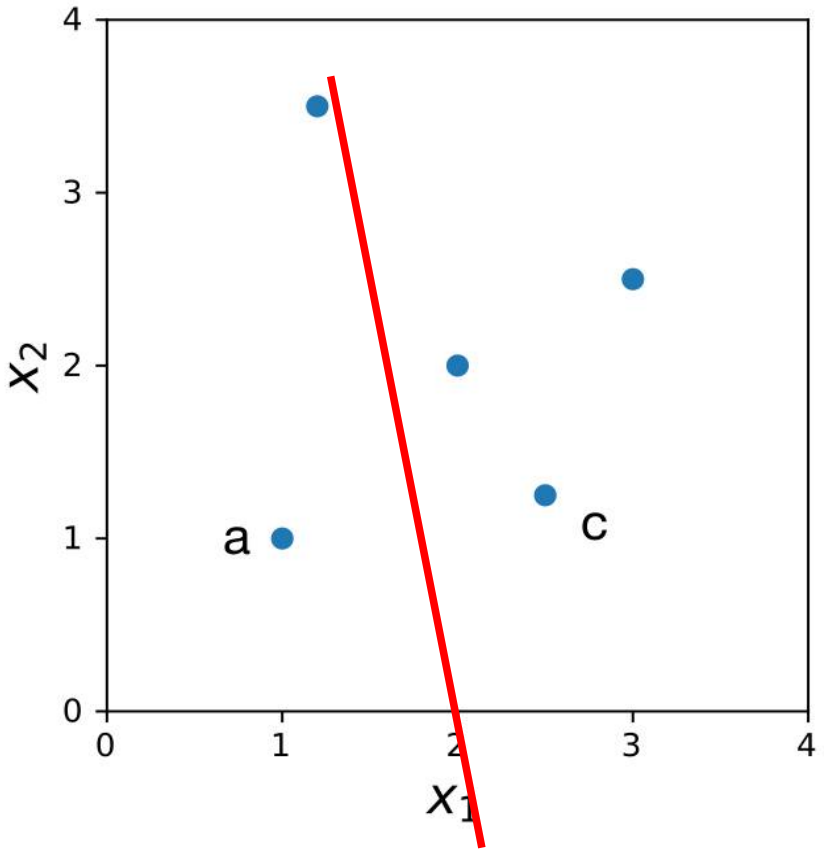


Assuming
1-nearest neighbor

Line is equi-distant
from **a** and **b**

How does it look like?

Decision Boundary between points (a) and (c)

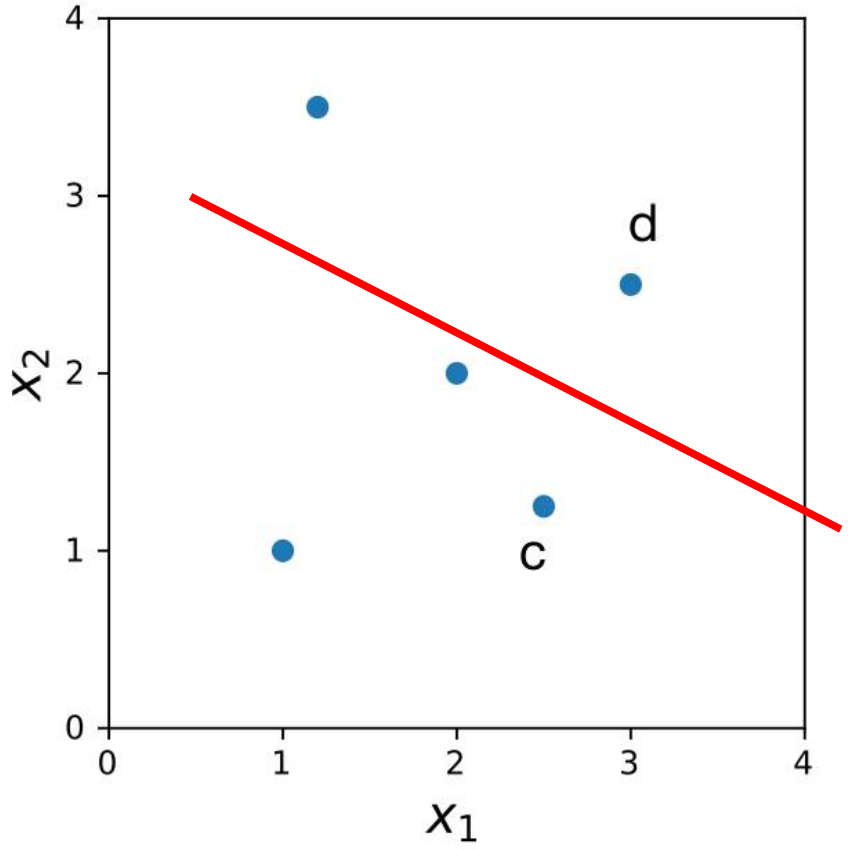


Assuming
1-nearest neighbor

Line is equi-distant
from **a** and **c**

How does it look like?

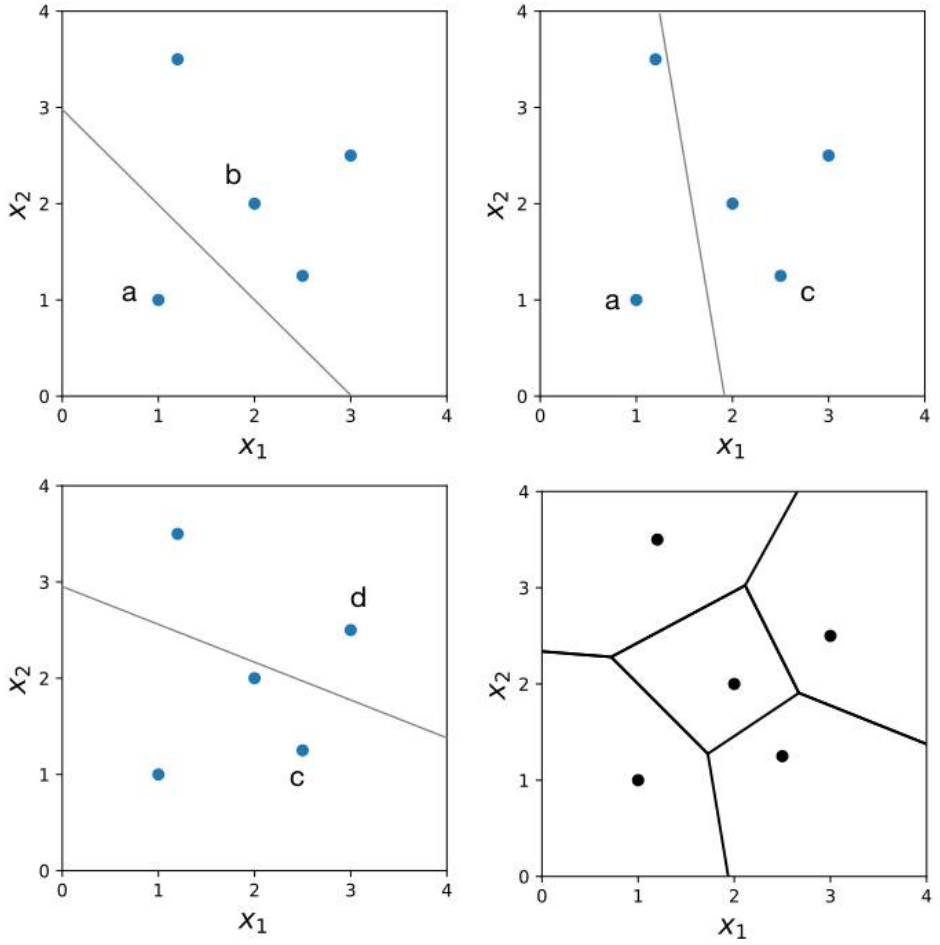
Decision Boundary between points (a) and (d)



Assuming
1-nearest neighbor

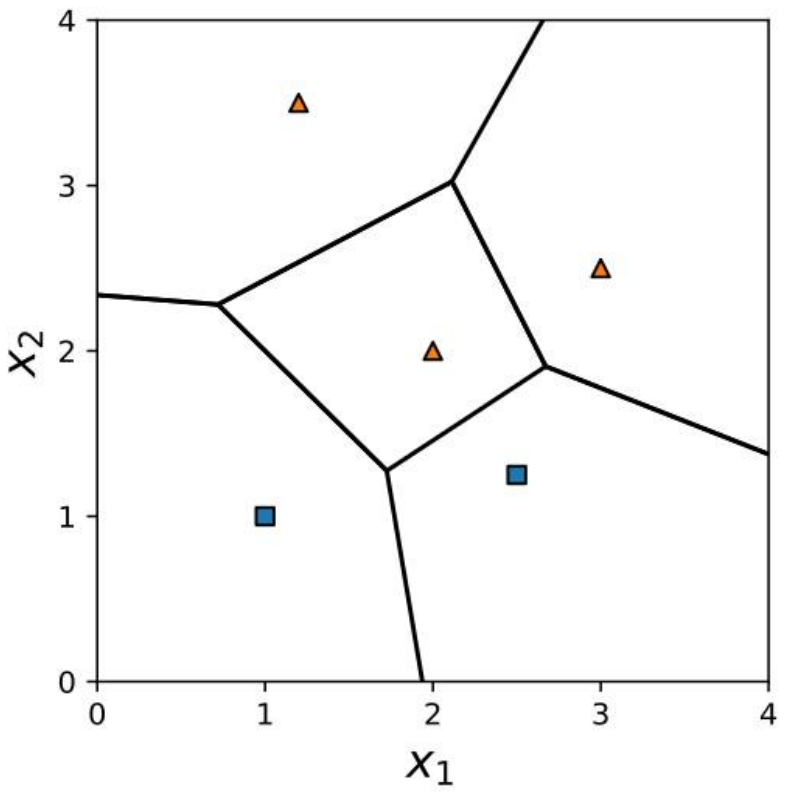
Line is equi-distant
from **c** and **d**

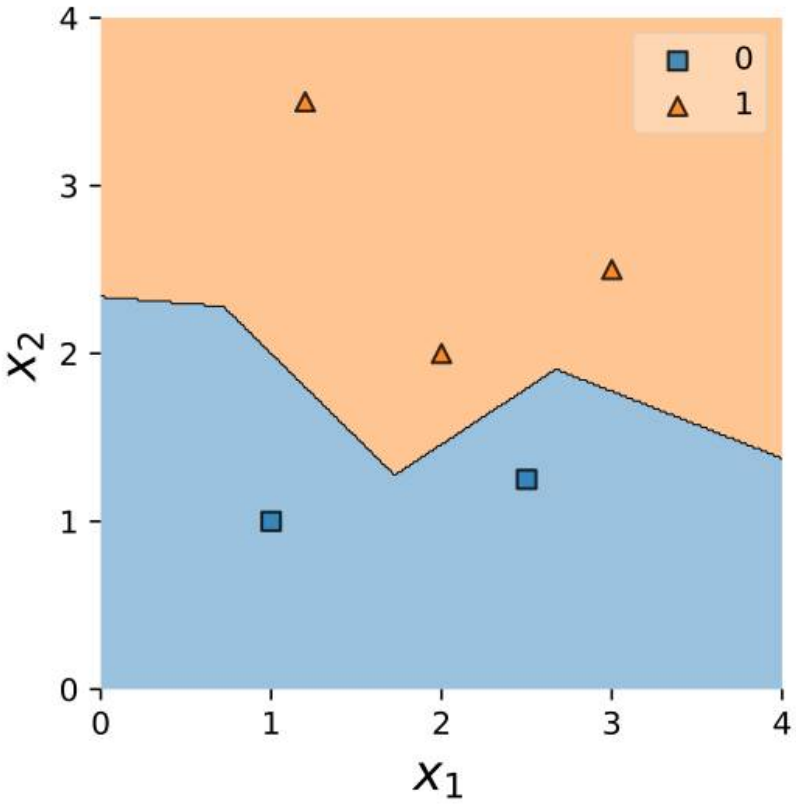
Decision boundary for 1-NN



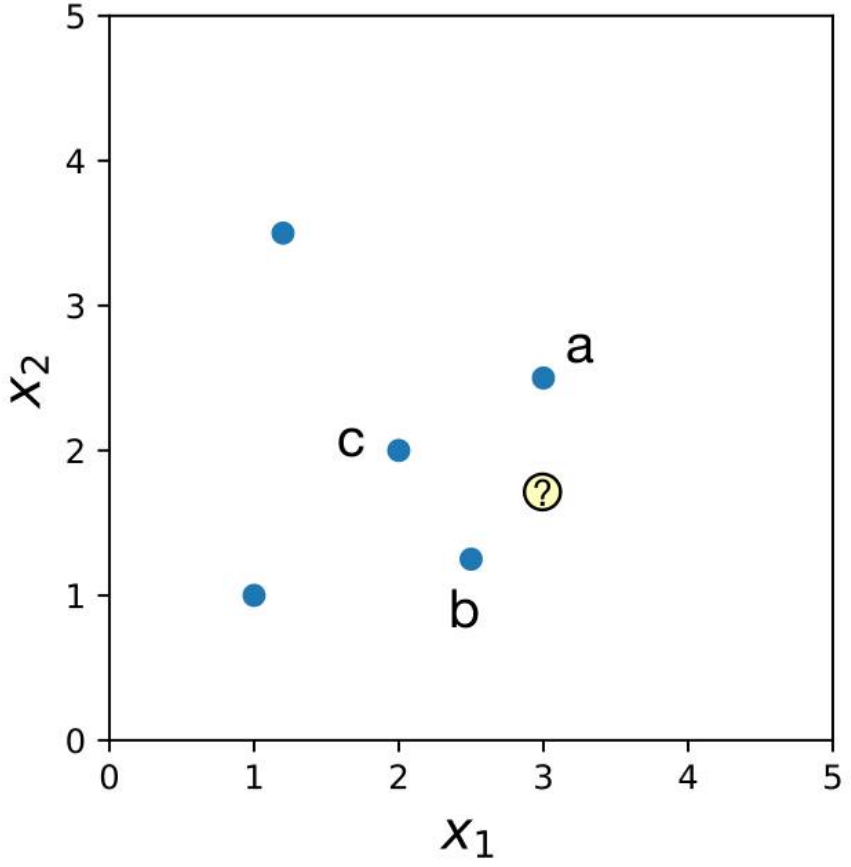
Voronoi Diagram

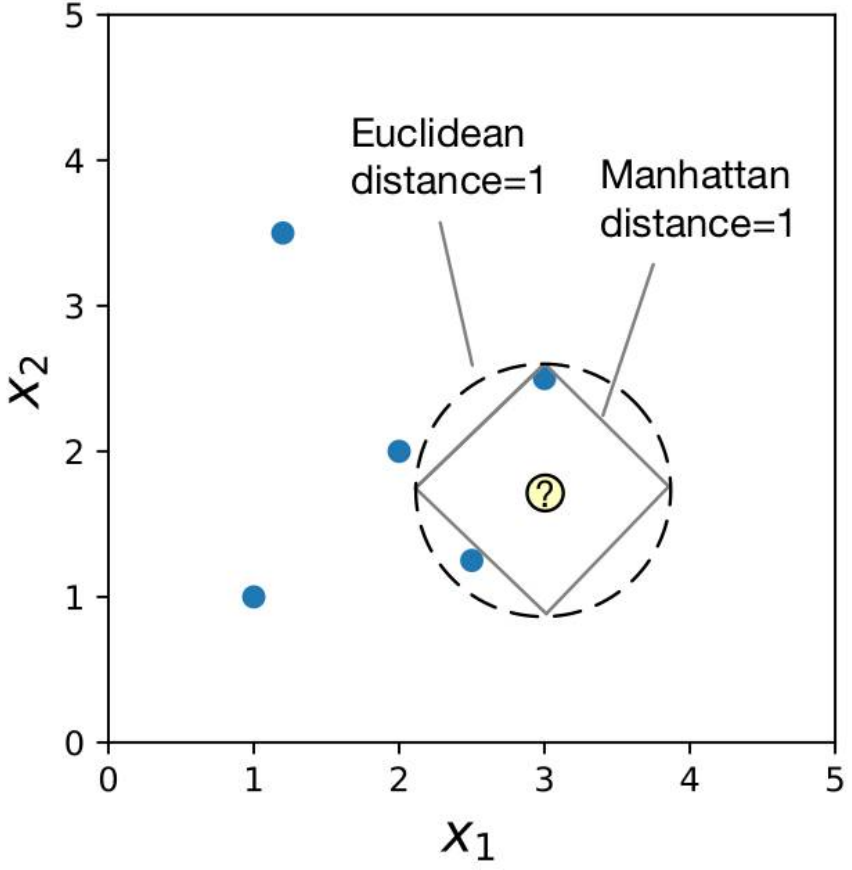
Decision boundary for 1-NN: Considering Class labels





“**Closeness**” depends on the Distance measure





Some Common Continuous Distance Measures

Euclidean p = 2

Also called TaxiCab Distance

Manhattan p = 1

Minkowski: $d(\mathbf{x}^{[a]}, \mathbf{x}^{[b]}) = \left[\sum_{j=1}^{m} \left(\left| x_j^{[a]} - x_j^{[b]} \right| \right)^p \right]^{\frac{1}{p}}$

Mahalanobis

Cosine similarity

...

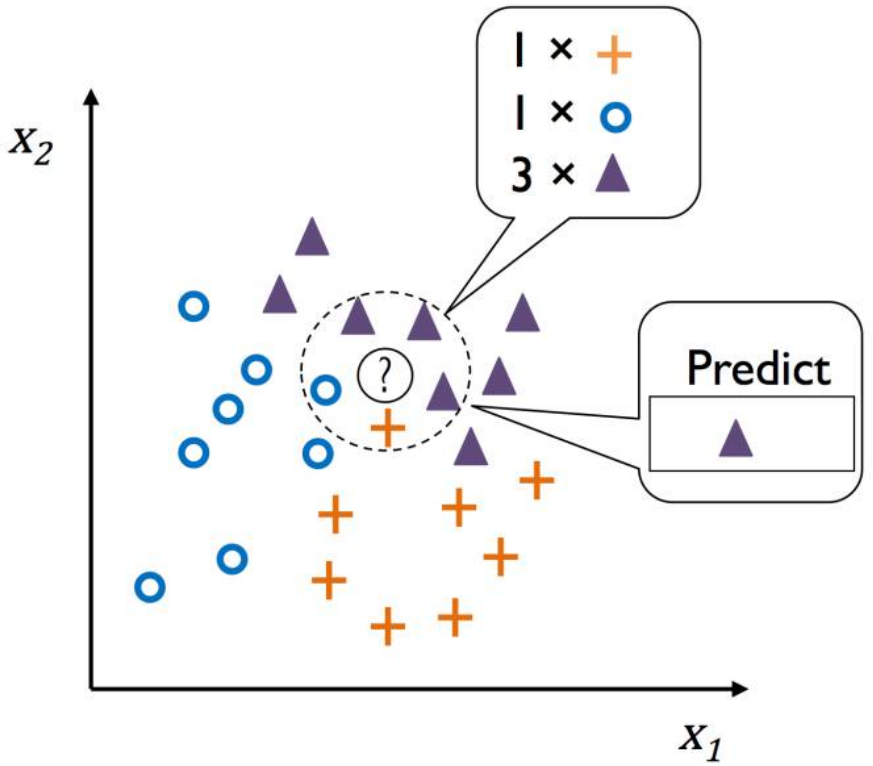
Some Discrete Distance Measures

Hamming distance: $d(\mathbf{x}^{[a]}, \mathbf{x}^{[b]}) = \sum_{j=1}^{m} \left| x_j^{[a]} - x_j^{[b]} \right|$ where $x_j \in \{0,1\}$

Jaccard/Tanimoto similarity:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

k – Nearest Neighbors



k = 5

Naive Nearest Neighbor: k-NN

$\mathcal{O}(k\,n)$

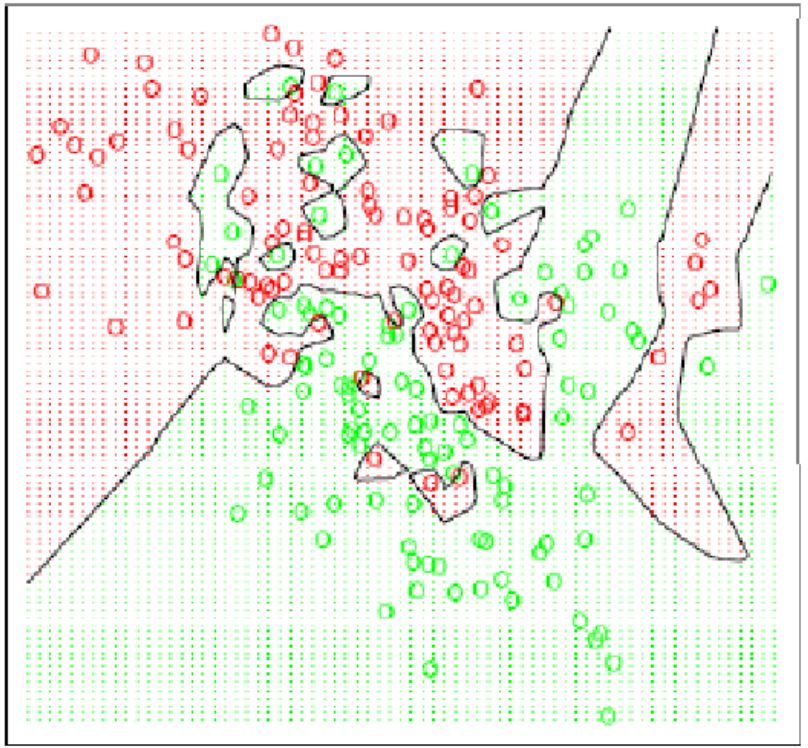
$\mathcal{D}_k := \{\}$

while $|\mathcal{D}_k| < k$:

- `closest_distance` $:= \infty$
- for $i = 1, ..., n, \quad \forall i \notin \mathcal{D}_k$:
  - `current_distance` $:= d(\mathbf{x}^{[i]}, \mathbf{x}^{[q]})$
  - if `current_distance` $<$ `closest_distance`:
    - `closest_distance` $:=$ `current_distance`
    - `closest_point` $:= \mathbf{x}^{[i]}$
- add `closest_point` to $\mathcal{D}_k$

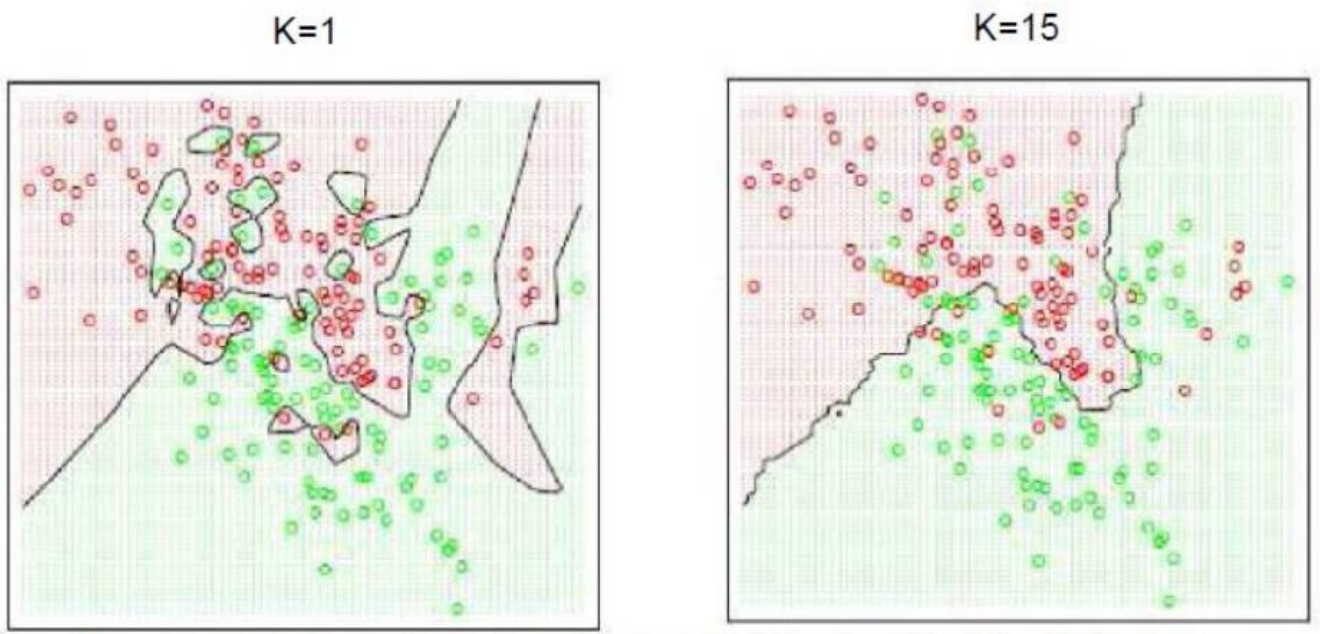
Decision Boundaries: k-NN

- With large number of examples and possible noise in the labels, the decision boundary can become highy irregular and distorted.
- "**Overfitting**" problem



Effect of k

- Larger **k** produces smoother boundary effect.
- When **k == N**, always predict the majority class.



Figures from Hastie, Tibshirani and Friedman (Elements of Statistical Learning)

Effect of k

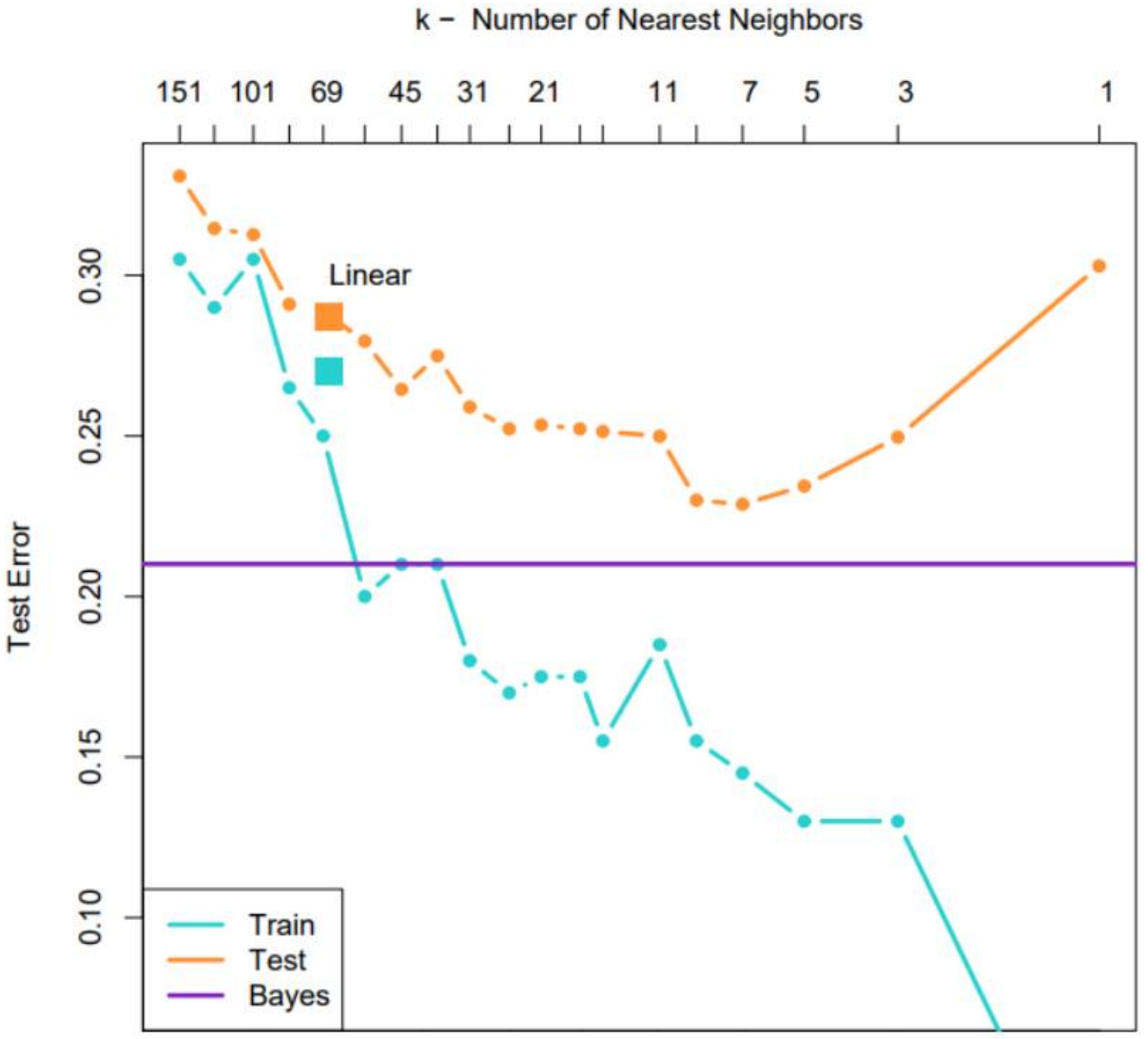


Image source: Elements of Statistical Learning”

How do we choose k ?

- Larger *k* may lead to better performance. But if we set k too large we may end up looking at samples that are not neighbors (i.e., are far away from the query point)
- We can use validation set / cross-validation to find k.
- Rule of thumb is *k* < *sqrt(n)*, where *n* is the number of training examples.

Special Cases of Minkowski Distance

- Consider the Minkowski Distance
  - 3 cases:
    - p = 1 : L_1 norm or Manhattan Dist / Taxicab or Cityblock dist
    - p = 2 : L_2 or Euclidean Dist
    - p --> ∞ : L_{max} or L_∞ norm / max difference of 2 dimensions

Curse of Dimensionality

- The curse of dimensionality is a term introduced by Bellman to describe the problem caused by the exponential increase in volume associated with adding extra dimensions to Euclidean space (Bellman, 1957).
  - As the dimensionality (i.e., the number of variables) of a space grows, data points become so spread out that the ideas of *distance* and *density* become murky.
  - This is simply due to the incredible spatial increase that comes from adding an additional dimension.
  - As the number of features or dimensions grows, the amount of data we need to generalize accurately grows exponentially.

Curse of Dimensionality

- Suppose a dataset in distributed uniformly in the space $\mathbf{R^d}$. A point T in this space can have k neighbours out of n points that are roughly k/n distance away from it. This distance can considered within a box of volume L^d . Now *L* can be estimated w.r.t. *d*.
- As *d* increases, *L* increases to fill up the entire space.

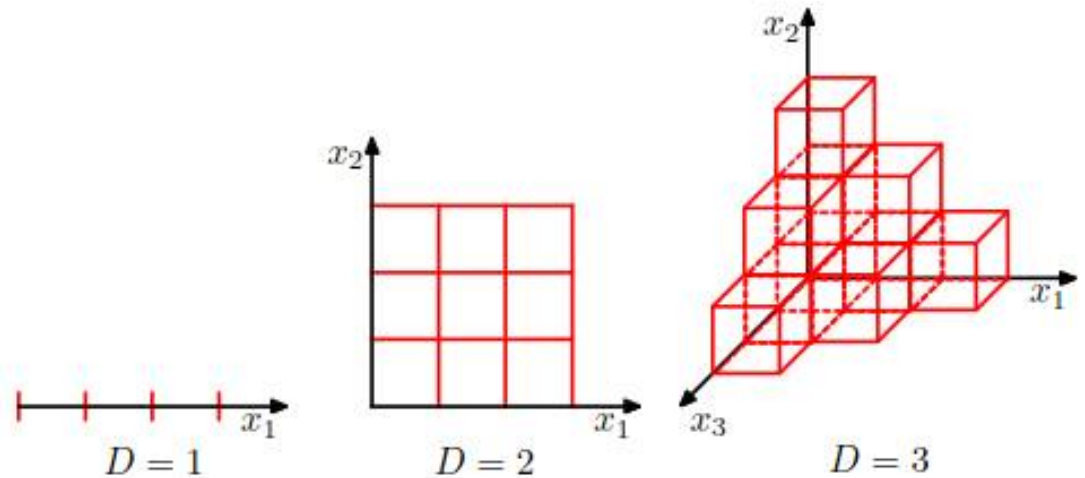
$L^d = k/n$

$L = (k/n)^{1/d}$

d	L
2	0.1
10	0.63
100	0.955
1000	0.9954

Curse of Dimensionality

- If we divide a region of a space into regular cells, then the number of such cells grows exponentially with the dimensionality of the space. The problem with an exponentially large number of cells is that we would need an exponentially large quantity of training data in order to ensure that the cells are not empty.



Pros and Cons

- Pros
  - Learning and implementation is extremely simple and Intuitive
  - Flexible decision boundaries
- Cons
  - It is only as good as its distance metric.
  - Irrelevant or correlated features have high impact and must be eliminated
  - Typically difficult to handle high dimensionality
  - Computational costs: memory and classification time computation

Points to Note: k - Nearest Neighbors

- Non-parametric approach. Does not make any underlying assumption regarding the distribution of the data.
- All the training examples are retained, and at evaluation time the “nearest” k neighbors are found and then averaged to produce the output.
- k is a hyperparameter.
- When k is too small, it **overfits** to the training data. As k gets larger, the classifier underfits (over-smooths) the data, resulting in the shrinkage of the two smaller regions.
- It does not scale. It becomes impractical for large datasets. Computational complexity increases.

Points to Note

- K-NN is simple but very powerful classification algorithm under certain assumptions.
- It classifies based on similarity/distance measure.
- Lazy learning
  - It does not “learn” until the test example is given
  - Whenever we have a new data to classify, we find its k-nearest neighbors from the training data.

References

- Kilian Weinberger Lecture 4:
  - https://www.youtube.com/watch?v=BbYV8UfMJSA&list=PLl8OlHZGYOQ7bkVbuRthEsaLr7bONzbXS&index=4
- Lectures by Sebastian Raschka
  - https://www.youtube.com/watch?v=-8ok7PuQEAk&list=PLTKMiZHVd_2KyGirGEvKlniaWeLOHhUF3&index=7

End of Lecture