Learning From Data
Lecture 17
Memory and Efficiency in Nearest Neighbor

Memory
Efficiency

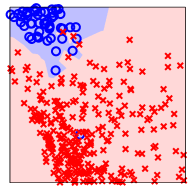
M. Magdon-Ismail
CSCI 4100/6100

RECAP: Similarity and Nearest Neighbor

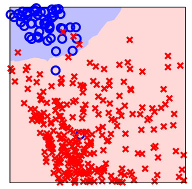
Similarity

$d(\mathbf{x}, \mathbf{x}') = \| \mathbf{x} - \mathbf{x}' \|$

1-NN rule



21-NN rule



1. Simple.
2. No training.
3. Near optimal E_{out}:
   $k \to \infty,\ k/N \to 0 \implies E_{\text{out}} \to E^*_{\text{out}}$.
4. Good ways to choose k:
   $k = 3$; $k = \left\lceil \sqrt{N} \right\rceil$; validation/cross validation.
5. Easy to justify classification to customer.
6. Can easily do multi-class.
7. Can easily adapt to regression or logistic regression

   $g(\mathbf{x}) = \dfrac{1}{k} \sum_{i=1}^{k} y_{[i]}(\mathbf{x})$ $\quad$ $g(\mathbf{x}) = \dfrac{1}{k} \sum_{i=1}^{k} [\![y_{[i]}(\mathbf{x}) = +1]\!]$
8. **Computationally demanding.**

Computational Demands of Nearest Neighbor

Memory.

Need to store all the data, $O(Nd)$ memory.

$N = 10^6$, $d = 100$, double precision $\approx$ 1GB

Finding the nearest neighbor of a test point.

Need to compute distance to every data point, $O(Nd)$.

$N = 10^6$, $d = 100$, 3GHz processor $\quad \approx$ 3ms (compute $g(\mathbf{x})$)

$\approx$ 1hr (compute CV error)

$>$ 1month (choose best k from among 1000 using CV)

Two Basic Approaches

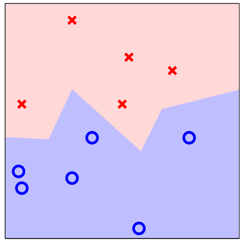
Reduce the amount of data.

The 5-year old does not remember every horse he has seen, only a few *representative* horses.

Store the data in a specialized data structure.

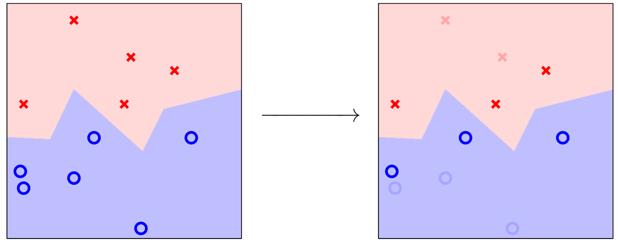
Ongoing research field to develop geometric data structures to make finding nearest neighbors fast.

Throw Away Irrelevant Data



$$k = 1$$

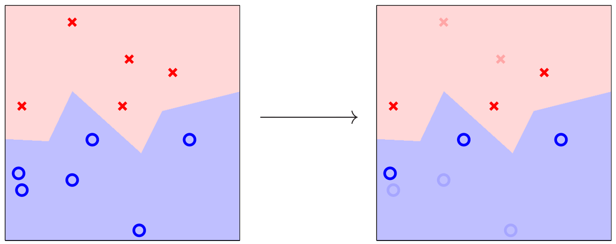
Decision Boundary Consistent



$g(\mathbf{x})$ unchanged

Does E_{in} change?
Does E_{out} change?

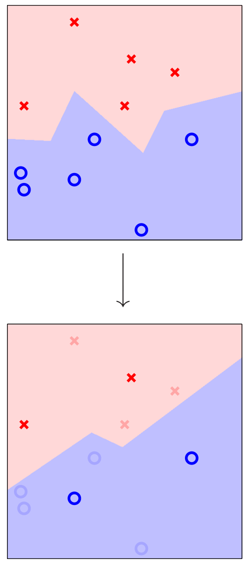
Decision Boundary Consistent



$g(\mathbf{x})$ unchanged

Does E_{in} change? No
Does E_{out} change? No

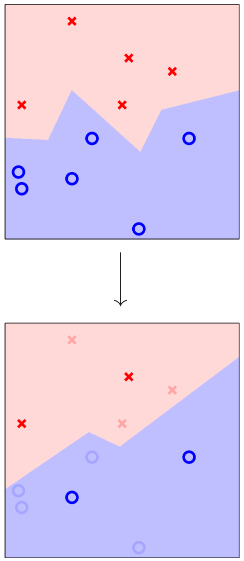
Training Set Consistent



$g(\mathbf{x}_n)$ unchanged

Does E_{in} change?
Does E_{out} change?

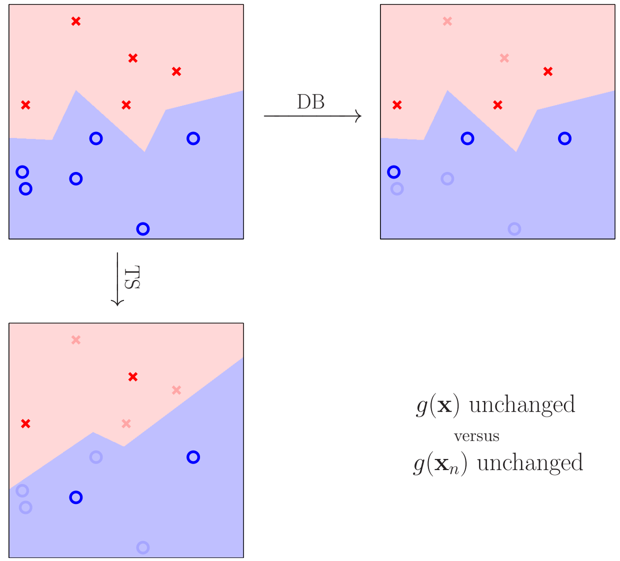
Training Set Consistent



$g(\mathbf{x}_n)$ unchanged

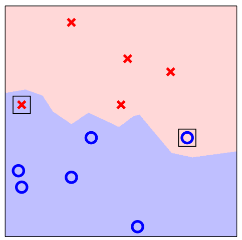
Does E_{in} change? No
Does E_{out} change? Usually

Decision Boundary Vs. Training Set Consistent



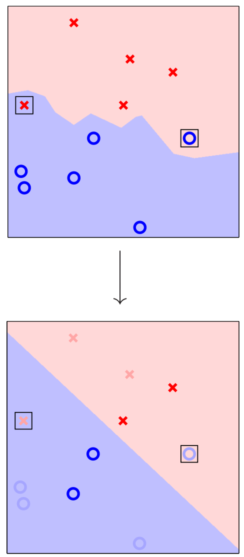
$g(\mathbf{x})$ unchanged
versus
$g(\mathbf{x}_n)$ unchanged

Consistent Does Not Mean $g(\mathbf{x}_n) = y_n$



$$k = 3$$

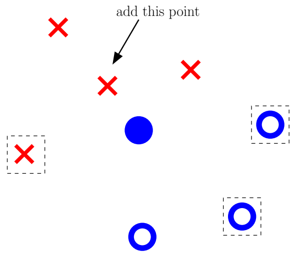
Training Set Consistent ($k = 3$)



$g(\mathbf{x}_n)$ unchanged

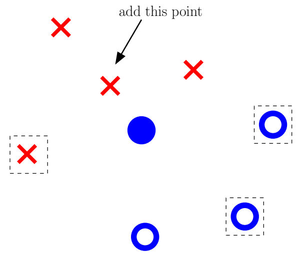
Does E_{in} change? No
Does E_{out} change? Usually

CNN: Condensed Nearest Neighbor $(k = 3)$



Consider the solid blue point:
  i. blue w.r.t. selected points
  ii. red w.r.t. $\mathcal{D}$

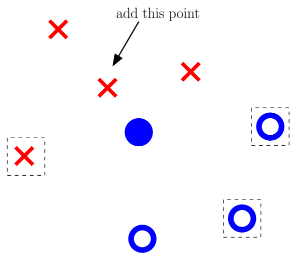
CNN: Condensed Nearest Neighbor



Consider the solid blue point:
  i. blue w.r.t. selected points
  ii. red w.r.t. $\mathcal{D}$

Add a red point:
  i. not already selected
  ii. closest to the inconsistent point

CNN: Condensed Nearest Neighbor



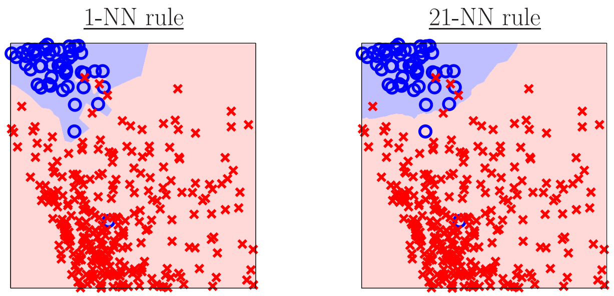
Consider the solid blue point:
  i. blue w.r.t. selected points
  ii. red w.r.t. $\mathcal{D}$

Add a red point:
  i. not already selected
  ii. closest to the inconsistent point

1. Randomly select k data points into $\mathcal{S}$.
2. Classify all data according to $\mathcal{S}$.
3. Let $\mathbf{x}_*$ be an inconsistent point and y_* its class w.r.t. $\mathcal{D}$.
4. Add the closest point to $\mathbf{x}_*$ not in $\mathcal{S}$ that has class y_*.
5. Iterate until $\mathcal{S}$ classifies all points consistently with $\mathcal{D}$.

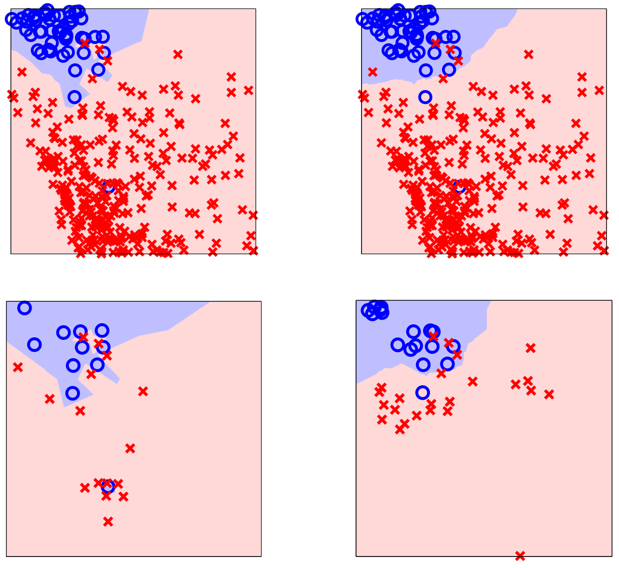
Minimum consistent set (MCS)? ← NP-hard

Nearest Neighbor on Digits Data



Condensing the Digits Data

1-NN rule　　　　21-NN rule



Finding the Nearest Neighbor

1. S_1, S_2 are 'clusters' with centers $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$ and radii r_1, r_2.
2. **[Branch]** Search S_1 first $\rightarrow \hat{\mathbf{x}}_{[1]}$.
3. The distance from $\mathbf{x}$ to any point in S_2 is at least

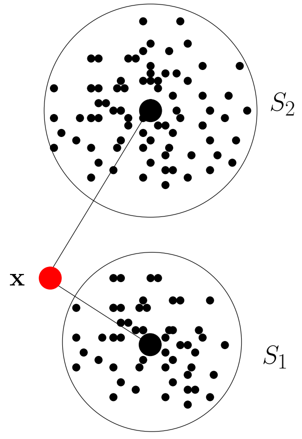
$$\|\mathbf{x} - \boldsymbol{\mu}_2\| - r_2$$

4. **[Bound]** So we are done if

$$\|\mathbf{x} - \hat{\mathbf{x}}_{[1]}\| \leq \|\mathbf{x} - \boldsymbol{\mu}_2\| - r_2$$

A *branch and bound* algorithm

Can be applied *recursively*



When Does the Bound Hold?

Bound condition: $\|\mathbf{x} - \hat{\mathbf{x}}_{[1]}\| \leq \|\mathbf{x} - \boldsymbol{\mu}_2\| - r_2$.

$$\|\mathbf{x} - \hat{\mathbf{x}}_{[1]}\| \leq \|\mathbf{x} - \boldsymbol{\mu}_1\| + r_1$$

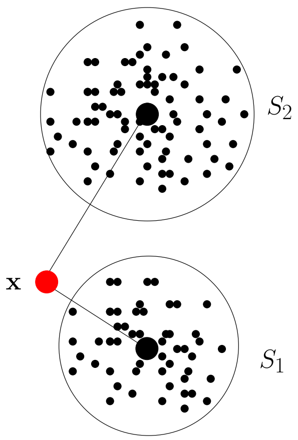
So, it suffices that

$$r_1 + r_2 \leq \|\mathbf{x} - \boldsymbol{\mu}_2\| - \|\mathbf{x} - \boldsymbol{\mu}_1\|.$$

$\|\mathbf{x} - \boldsymbol{\mu}_1\| \approx 0$ means $\|\mathbf{x} - \boldsymbol{\mu}_2\| \approx \|\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1\|$.

It suffices that

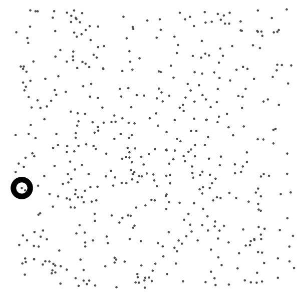
$$r_1 + r_2 \leq \|\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1\|.$$



within cluster spread should be less than *between cluster spread*

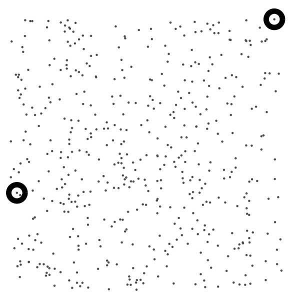
Finding Clusters – Lloyd's Algorithm

1. Pick well separated centers for each cluster.



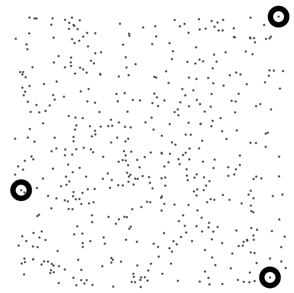
Finding Clusters – Lloyd's Algorithm

1. Pick well separated centers for each cluster.



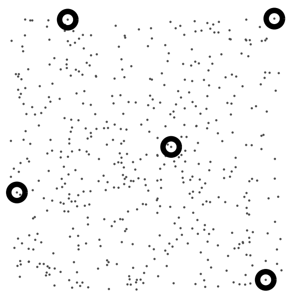
Finding Clusters – Lloyd's Algorithm

1. Pick well separated centers for each cluster.



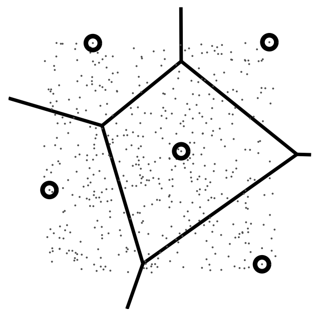
Finding Clusters – Lloyd's Algorithm

1. Pick well separated centers for each cluster.



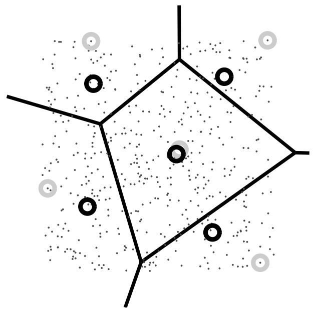
Finding Clusters – Lloyd's Algorithm

1. Pick well separated centers for each cluster.
2. Compute Voronoi regions as the clusters.



Finding Clusters – Lloyd's Algorithm

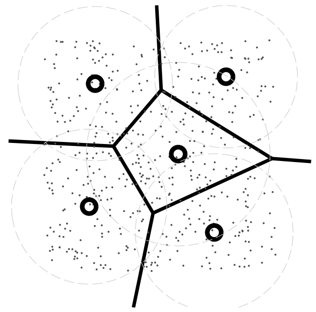
1. Pick well separated centers for each cluster.
2. Compute Voronoi regions as the clusters.
3. Update the Centers.



Finding Clusters – Lloyd's Algorithm

1. Pick well separated centers for each cluster.
2. Compute Voronoi regions as the clusters.
3. Update the Centers.
4. Update the Voronoi regions.
5. Compute centers and radii:

$$\boldsymbol{\mu}_j = \frac{1}{|S_j|}\sum_{\mathbf{x}_n \in S_j} \mathbf{x}_n; \qquad r_j = \max_{\mathbf{x}_n \in S_j} \|\mathbf{x}_n - \boldsymbol{\mu}_j\|.$$



Radial Basis Functions (RBF)

k-Nearest Neighbor: Only considers k-nearest neighbors.

each neighbor has equal weight

What about using *all* data to compute $g(\mathbf{x})$?

RBF: Use all data.

data further away from $\mathbf{x}$ have less weight.