Machine Learning from Data

Lecture 17: Spring 2021

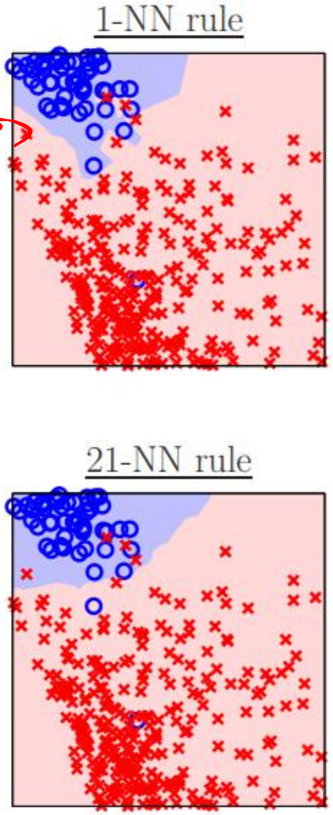
Today’s Lecture

- Memory and Efficiency of Nearest Neighbor

Similarity and Nearest Neighbor

Similarity

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



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:

$$k = 3;\ k = \left\lceil \sqrt{N} \right\rceil;\ \text{validation/cross validation.}$$

5. Easy to justify classification to customer.
6. Can easily do multi-class.
7. Can easily adapt to regression or logistic regression

$$g(\mathbf{x}) = \frac{1}{k}\sum_{i=1}^{k} y_{[i]}(\mathbf{x}) \qquad g(\mathbf{x}) = \frac{1}{k}\sum_{i=1}^{k} [\![y_{[i]}(\mathbf{x}) = +1]\!]$$

8. **Computationally demanding.**

Computational Demands of Nearest Neighbor

1 **Memory.**

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

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

computational.

2 **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 $\approx$ 3ms (compute $g(\mathbf{x})$)

$\approx$ 1hr (compute CV error)

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

Two Basic Approaches

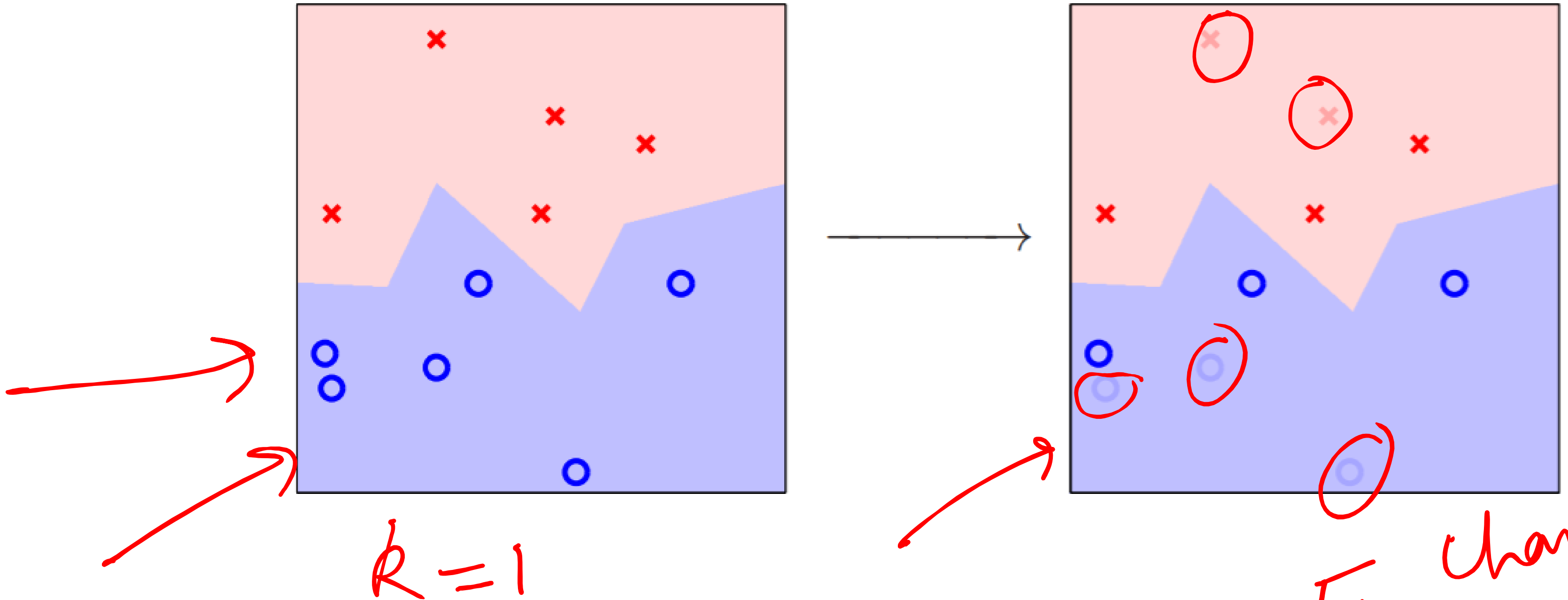
Reduce the amount of data. → Story → Memory

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.

Decision Boundary Consistent



$k=1$

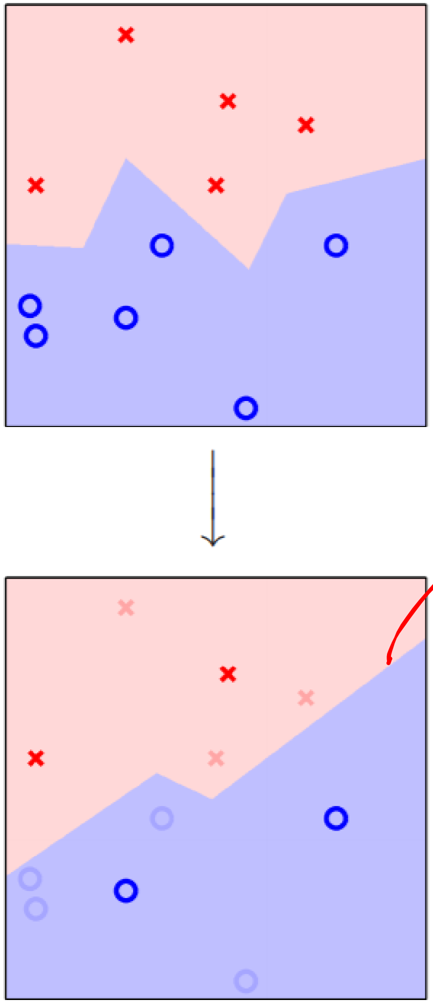
Condensed

E_{in} change? No

E_{out} change? No

$g(\mathbf{x})$ unchanged

Training Set Consistent



g

hypothesis

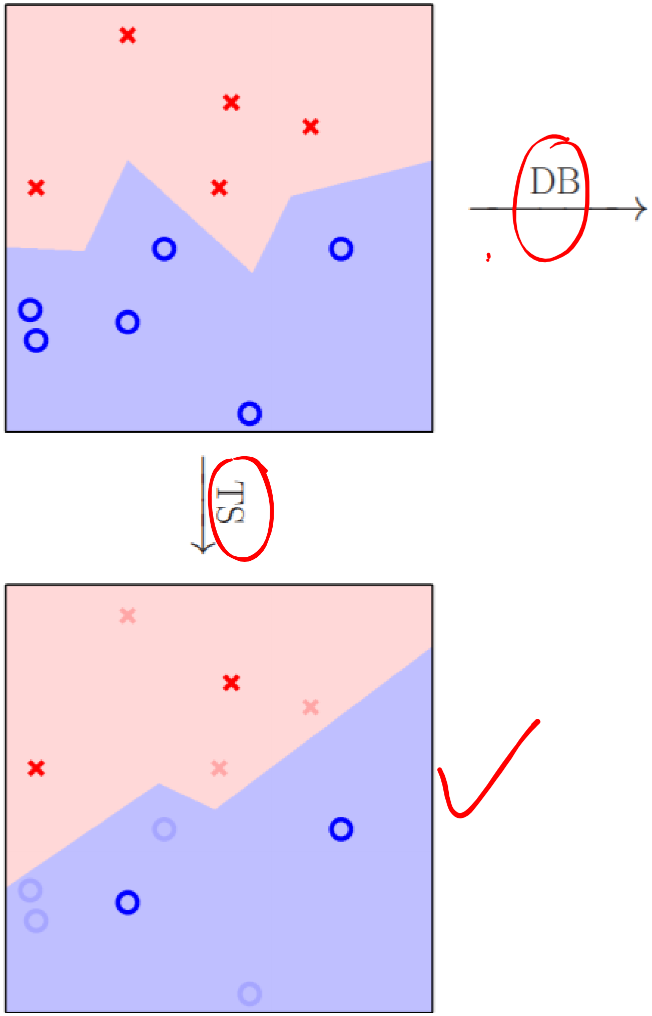
g^{-}

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

$E_{in} \rightarrow$ No.

$E_{out} \rightarrow$ hypothesis has changed so E_{out} has changed!

Decision Boundary Vs. Training Set Consistent



$E_{in} = 0$

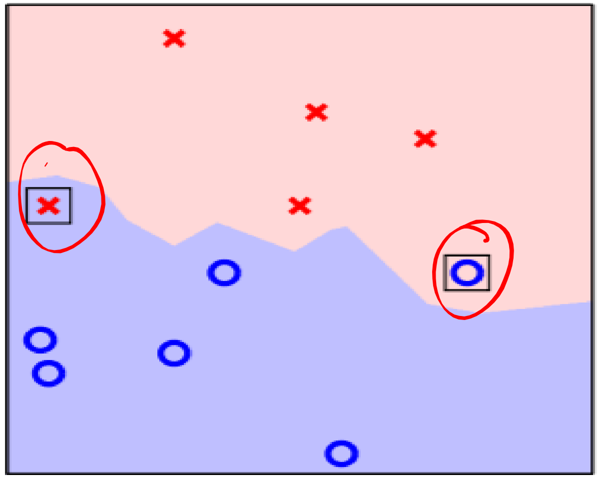
if $k = 3$

$g(\mathbf{x})$ unchanged

versus

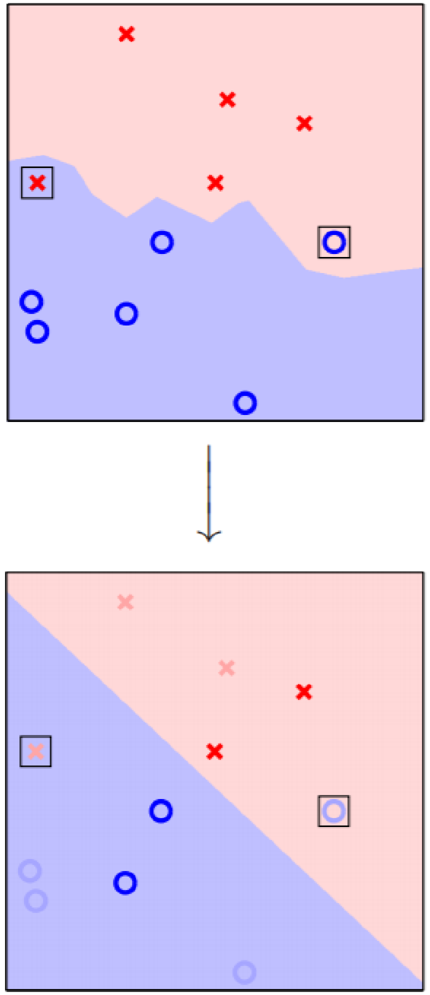
$g(\mathbf{x}_n)$ unchanged

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



$k = 3$

Training Set Consistent $(k = 3)$



$E_{in} \neq 0$

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

Condensed Nearest Neighbor (CNN)

Take data → condense it → NN, $k = 3$

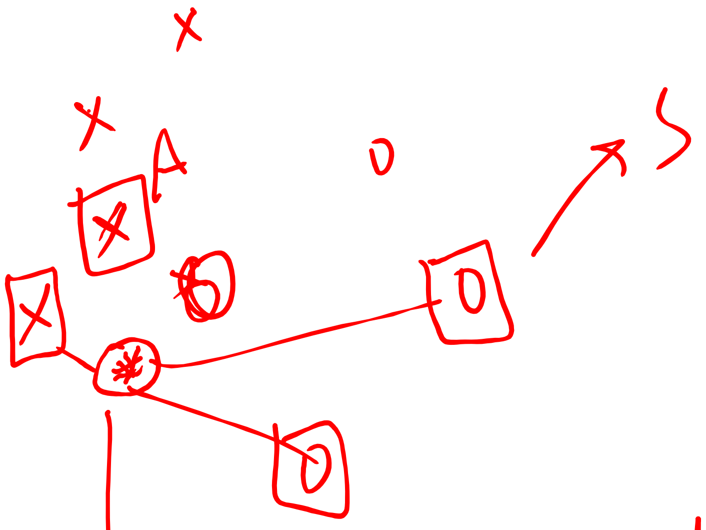
$$\underline{\underline{D}} = \begin{Bmatrix} x_1 & x_2 & \dots & x_N \\ y_1 & y_2 & \dots & y_N \end{Bmatrix}$$

S (working set)

$g_D(x)$

$\hat{y}_1 \quad \hat{y}_2 \quad \hat{y}_3 \; \dots \; \hat{y}_N$

$\hat{y}_i = g_D(x_i)$



$\underline{o \to +1}$

$g_s(*) = +1$

$g_D(*) = -1$

x_* reveals a training set inconsistency

$S \to g_S(x)$

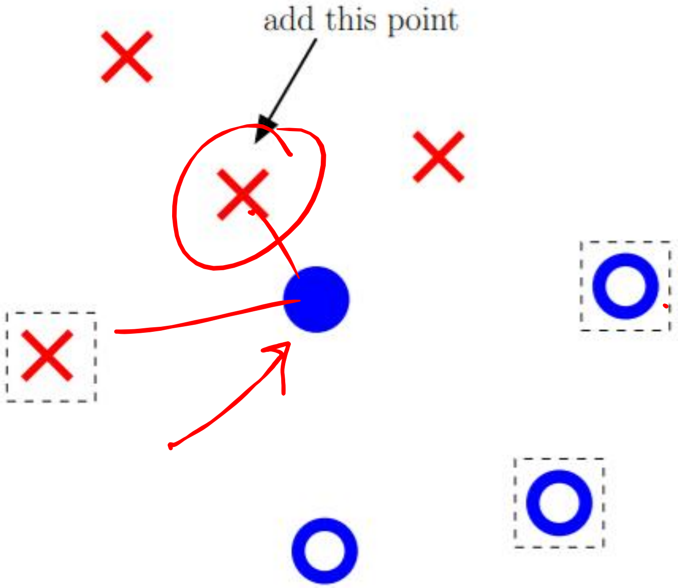
$g_D(x_i) = g_S(x_i)$

$S = \Phi$, $k = 3$

1. If S is TS consistent $\longrightarrow$ we are done
2. $x_*(y_*) \longrightarrow g_D(x_*) \neq g_S(x_*)$

   ↳ Add a point to your S
3. Add this point
4. Repeat these steps $\longrightarrow$ TS consistent.

CNN: Condensed Nearest Neighbor



Consider the solid blue point:

i. blue w.r.t. selected points

ii. red w.r.t. $\mathcal{D}$

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 $\mathcal{D}$.

Q1. How do we know such a point exists?

Q2. " " " " that this point is in D?

→ Yes

Q3. Algorithm stops or not?

N steps.

Theorem (1) Algorithm is well defined.

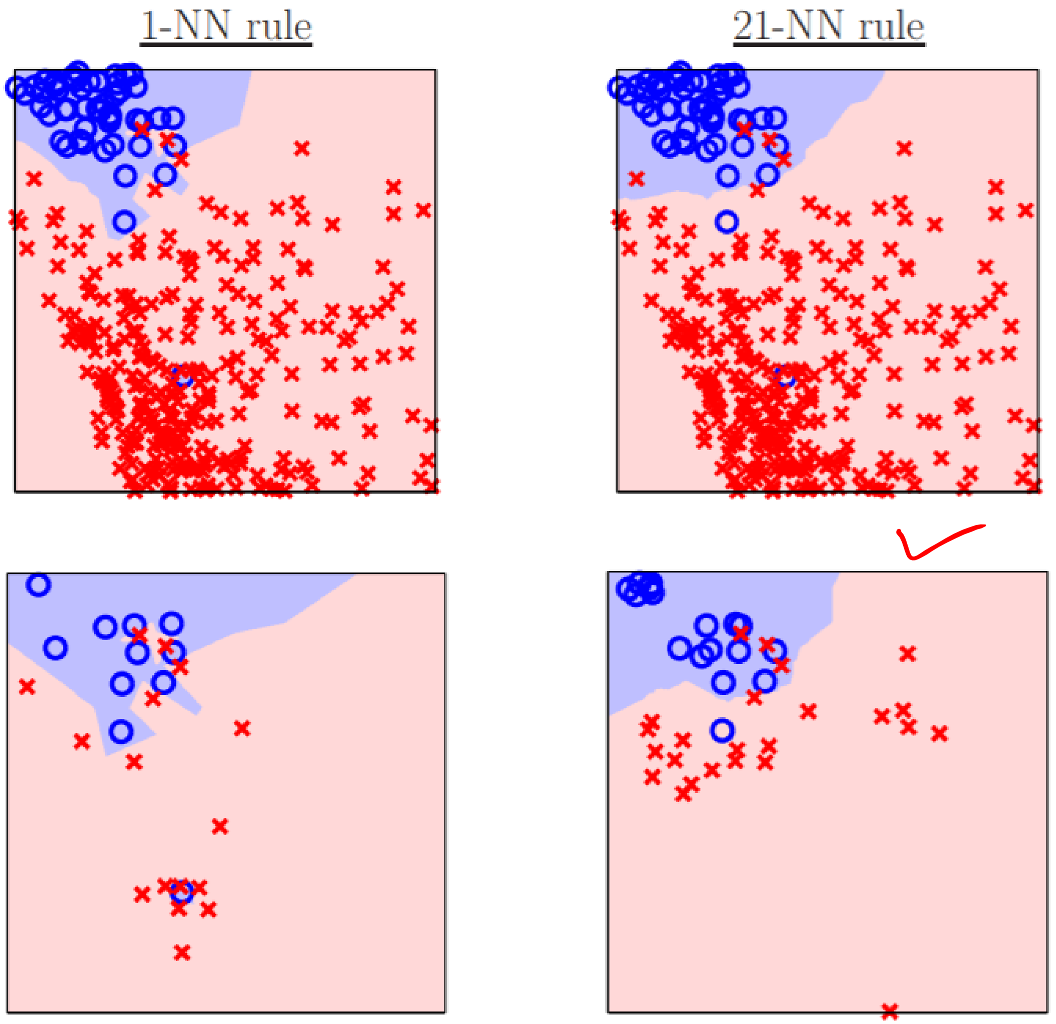
2) At most N steps.

3) On termination S must be in TS consistent.

Condensing the Digits Data

Original →

CNN →



TS

↓

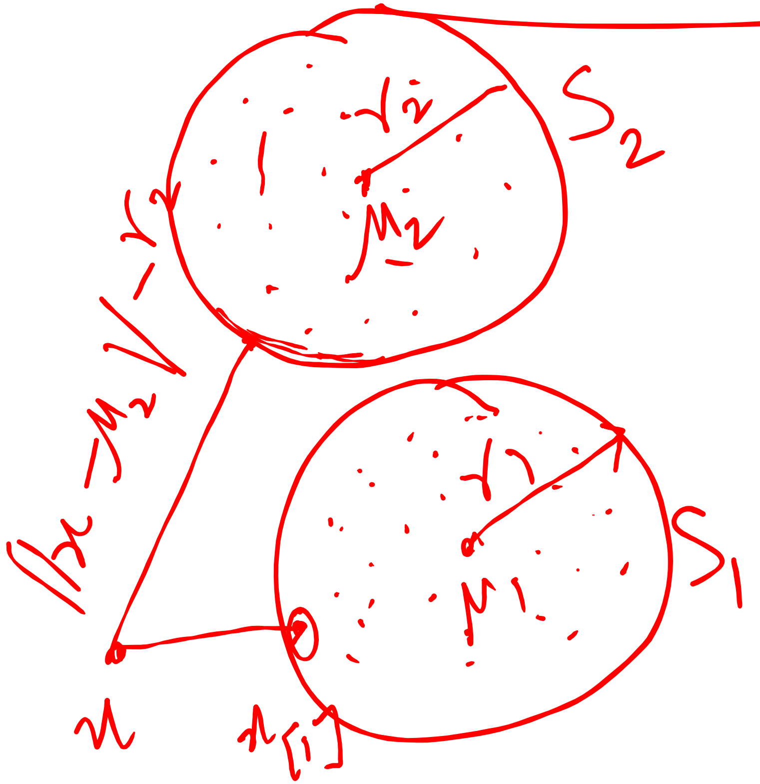
smallest subset?

NP-hard

K, N

Algorithmic efficiency → finding NN (Quickly)

Branch & Bound



1) Chose S_1 → Branch step

2) BOUND CRITERIA

If $\underbrace{\|x-\mu_2\| - r_2}$ $\geq \|x - \hat{x}_{[1]}\|$

No need to search S_2

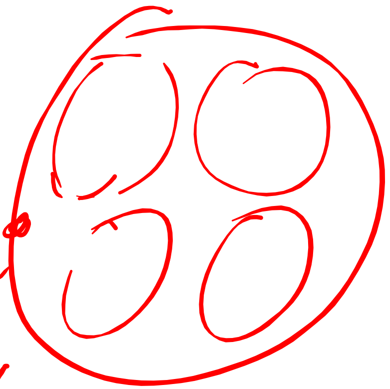
Observations

i) Produces the NN (guaranteed).

ii) Recursively

iii) Worst case $O(N)$

Relax → w.h.p → logarithmic x^{n}



Eout

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.

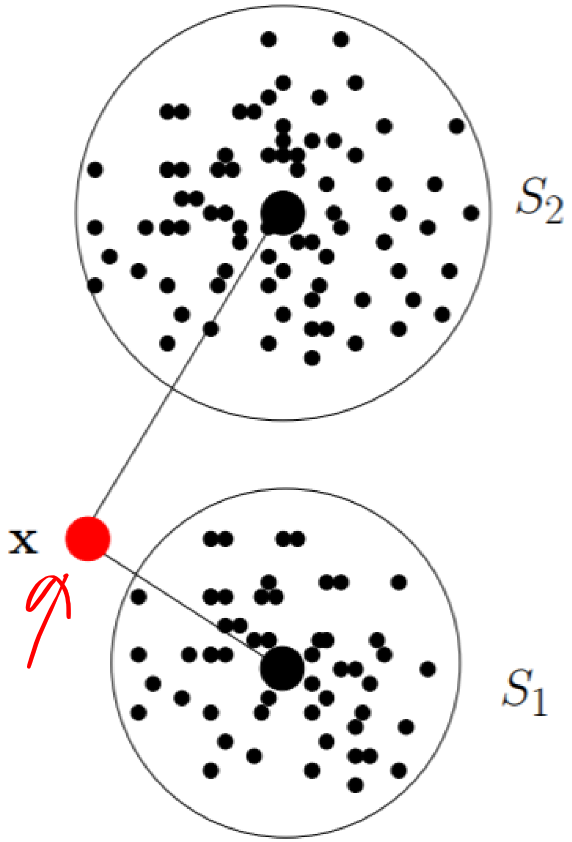
2. **[Branch]** Search S_1 first $\rightarrow \hat{\mathbf{x}}_{[1]}$.

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

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

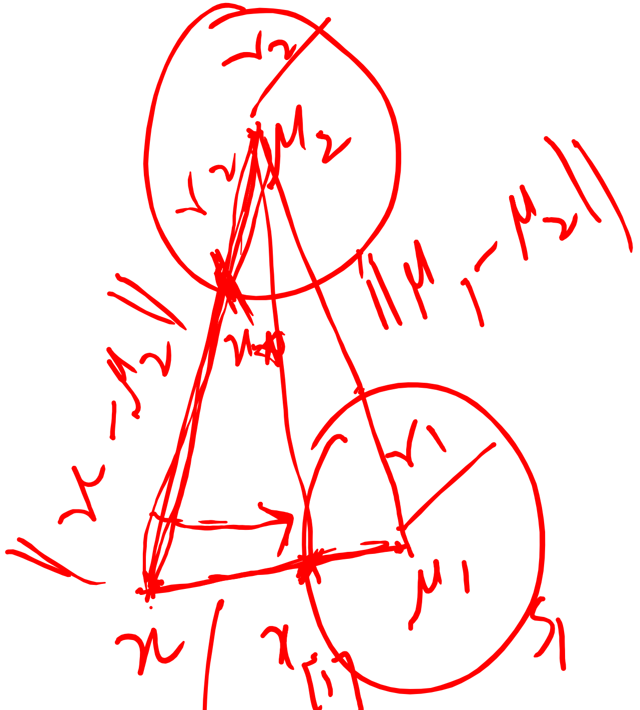


A *branch and bound* algorithm

Can be applied *recursively*

When do we gain? $r_1 + r_2 \to$ Inter, Intra

$$\|x - \hat{x}_{[i]}\| \le \underbrace{\|x - \mu_1\| + r_1}_{\text{any point in cluster 1}}$$



$$\|x - \mu_1\| + r_1 \le \|x - \mu_2\| - r_2$$

$\to$ Gain! $\to$ Sufficient condition.

$$\|x - \mu_1\| + r_1 \le \|x - \mu_2\| - r_2$$

$$r_1 + r_2 \le \|x - \mu_2\| - \underbrace{\|x - \mu_1\|}_{\text{small}}$$

$$\simeq \|\mu_1 - \mu_2\|$$

$$\|x - x_{[i]}\|$$

$$\boxed{r_1 + r_2 < \|\mu_1 - \mu_2\|}$$

When Does the Bound Hold?

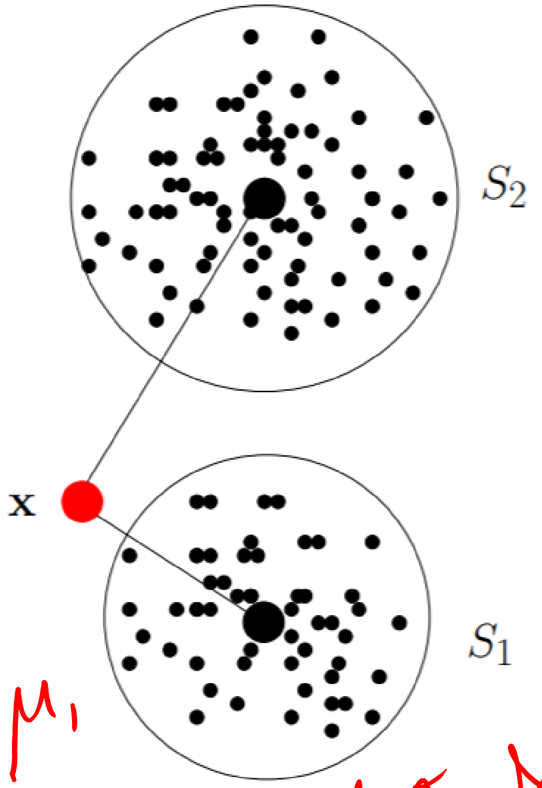
Bound condition: $\|\mathbf{x} - \hat{\mathbf{x}}_{[1]}\| \leq \|\mathbf{x} - \boldsymbol{\mu}_2\| - r_2$.

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

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



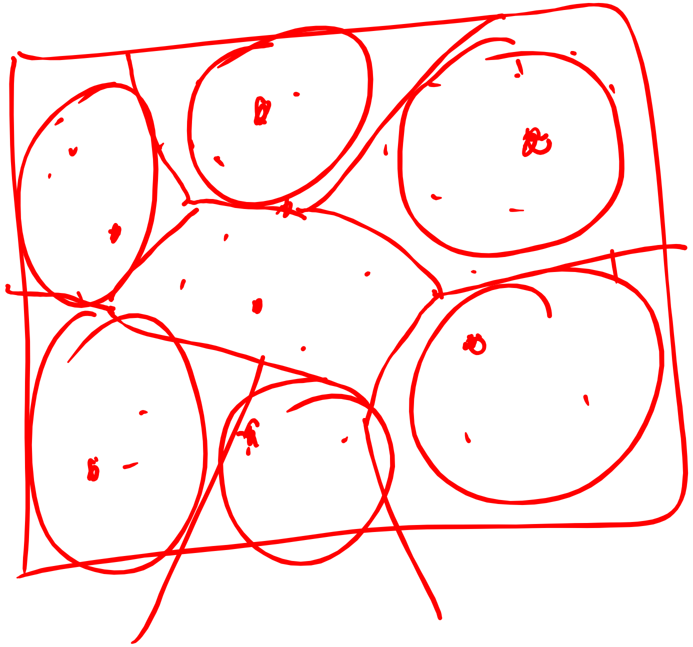
It suffices that $\boxed{r_1 + r_2 \leq \|\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1\|.}$

$r_1 + r_2 << \mu_2 - \mu_1$

tight & well separated.

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

Lloyd's Algo ✓



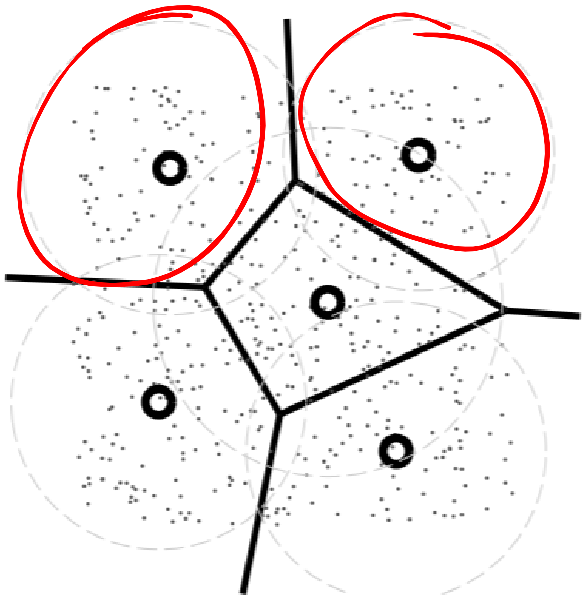
1) Pick center randomly.
2) Pick next as far as possible, voronoi.
3) Update the center to the actual center

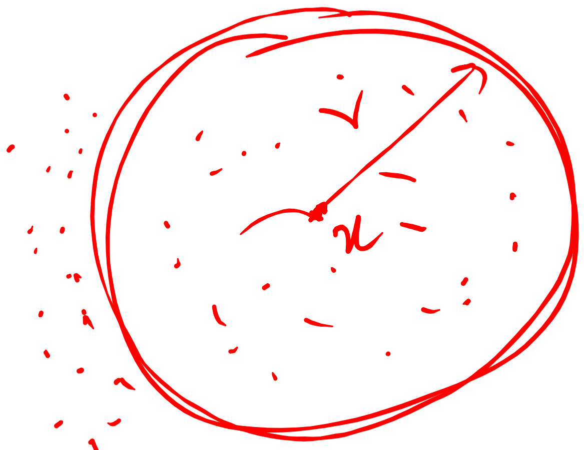
$$r_1 + r_2 \ll \|\mu_1 - \mu_2\|$$

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:

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





Bad $\rightarrow$ Weights

Radial Basis functions.

Thanks!