Multi-resolution Instance-Based Learning

Kan Deng kdeng@cs.cmu.edu

Andrew W. Moore awm@cs.cmu.edu

The Robotics Institute
School of Computer Science
Carnegie Mellon University

This Talk,

How can we make

Kernel Regression

less expensive?

Regression Problem

Real values, not Boolean !

Memory structure:

$\{ x_{11}, x_{12}, \ldots, x_{1k}, \quad y_1 \}, \ \{ x_{21}, x_{22}, \ldots, x_{2k}, \quad y_2 \}$

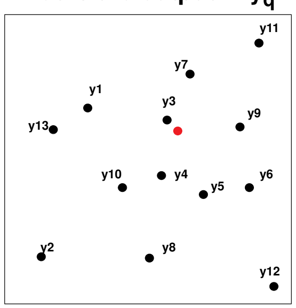
$\ldots \ldots, \{ x_{N1}, x_{N2}, \ldots, x_{Nk}, \quad y_N \}$ --- memory size: N

Query: Given input $\{ x_{q1}, x_{q2}, \ldots, x_{qk}, \}$, (doesn't appear in the memory)

what is the output y_q ?

Regression

not classification !

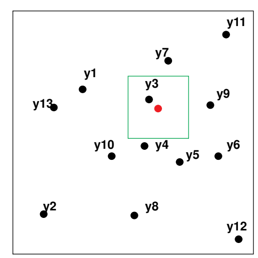


2-d input space

Two regression approaches: < 1 >

Nearest Neighbor: $y_q = \{ y \}_{nearest}$

nearest: the memory piece with the *smallest distance* to the query.
distance of m^{th} memory piece in *k*-d input space:



2-d input space

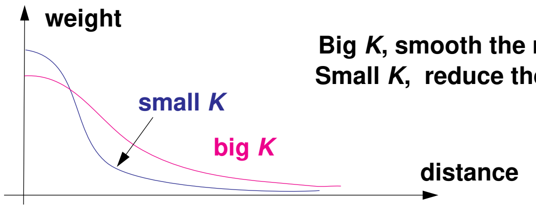
$$\sqrt{\sum_{i=1}^{k} \left(x_{mi} - x_{qi} \right)^2} \qquad m = 1, 2, \ldots, N$$

Two regression approaches: < 2 >

Kernel Regression:
reduce Nearest Neighbor's noise.

$$y_q = \frac{\sum_{i=1}^{N} w_i \cdot y_i}{\sum_{i=1}^{N} w_i}$$

i.e. weighted sum of all the outputs in the memory.



Big K, smooth the noise.
Small K, reduce the bias.

$$w_i = f\left(\text{distance}\left(\vec{x}_i, \vec{x}_q\right)\right) = exp\left(-\frac{\text{distance}\left(\vec{x}_i, \vec{x}_q\right)}{K^2}\right)$$

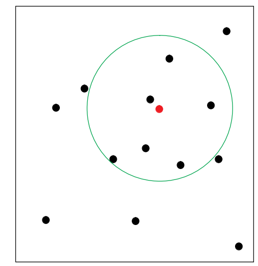
Drawback of Kernel Regression

Drawback:

too *expensive* to calculate all the individual weights

A Possible Solution:

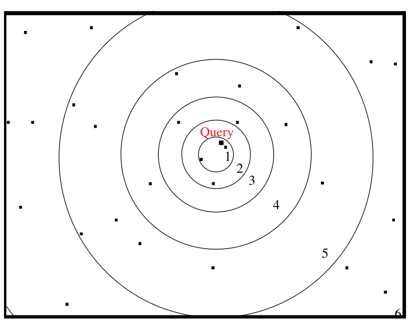
only use the *neighboring* memory pieces, ignore others outside.



2-d input space

To improve the efficiency of KR

Principle: partition the memory into several groups, each group share a common weight; instead of calculating the individual weights from every memory piece.



$$y_q = \frac{\sum_{i=1}^{N} w_i \cdot y_i}{\sum_{i=1}^{N} w_i} = \frac{\sum_{i=1}^{G} \bar{w}_i \left(\sum_{j=1}^{M_i} y_j\right)}{\sum_{i=1}^{G} \bar{w}_i M_i}$$

Construct a kd-tree

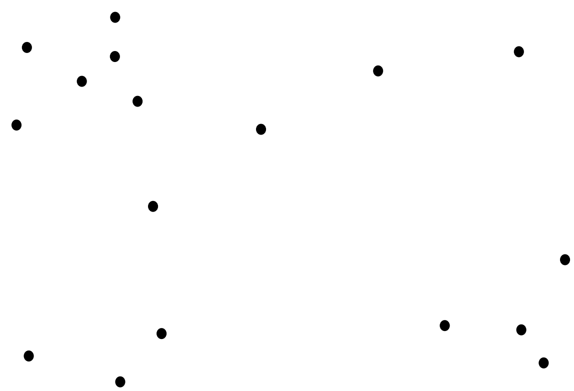
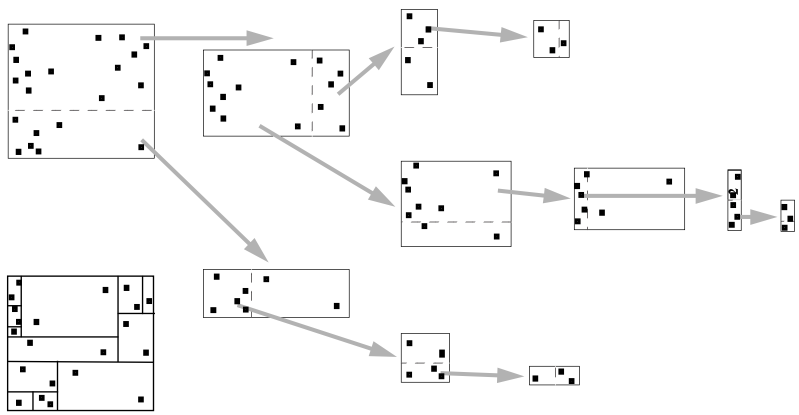


Illustration of *k*d-tree's structure



*K*d-tree for regression

1. Kernel Regression, $y_q = \dfrac{\sum_{i=1}^{N} w_i \cdot y_i}{\sum_{i=1}^{N} w_i}$, we need to calc $\sum_{i=1}^{N} w_i \cdot y_i$, $\sum_{i=1}^{N} w_i$

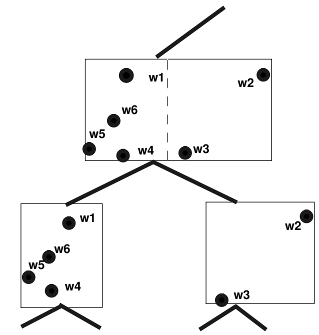
2. If the weights of the data points in one node are close enough,

   $\sum_{i=1}^{N} w_i \cdot y_i \approx \bar{w} \left(\sum_{i=1}^{N} y_j \right)$ $\qquad \sum_{i=1}^{N} w_i \approx \bar{w} N$

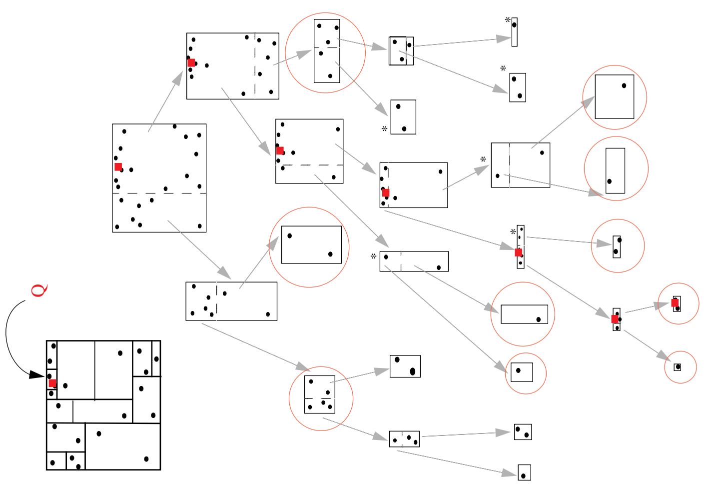
3. Travel the kd-tree top down,

   perform *cut-off* wheneven

   possible.



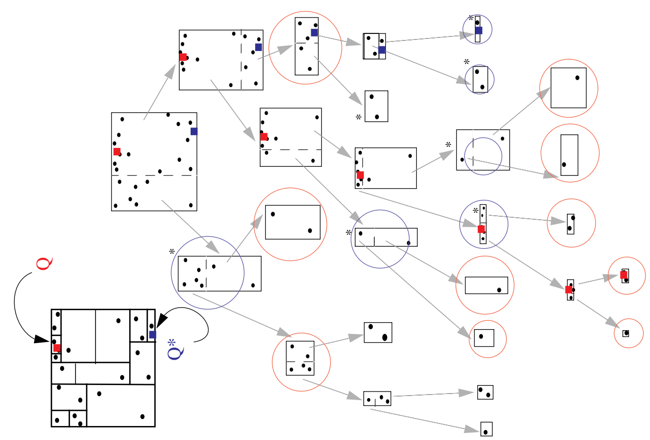
Multi-resolution Regression



$$y_q = \frac{\sum_{i=1}^{G} \bar{w}_i \cdot \left(\sum_{j=1}^{M_i} y_j \right) + \sum_{l=1}^{L} w_l \cdot y_l}{\sum_{i=1}^{G} \bar{w}_i \cdot M_i + \sum_{l=1}^{L} w_l}$$

*K*d-tree is powerful for regression

The same tree can be used for different regressions, with different *K*.

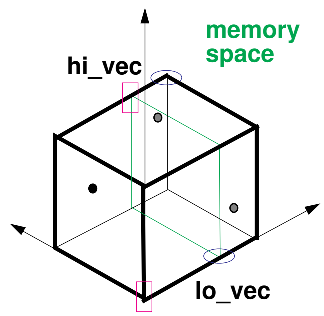


Kd-tree data structure

Each node in the tree represents a hyperbox.

It also contains,

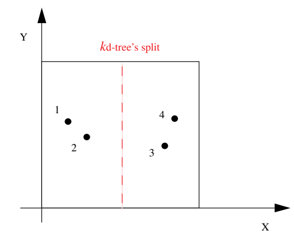
1. **The boundary of the box.**
2. **The child nodes of the box.**
3. **The sum of output values of all data points inside the box. i.e.** $\sum_{i=1}^{N} y_j$ **and *N*.**



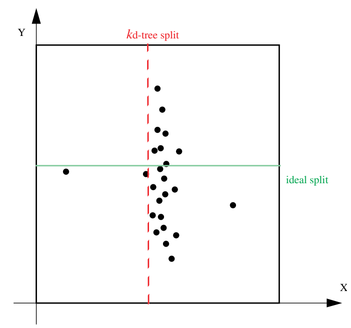
How to decide *split_d* and *split_v*

Goal:

make the memory pieces *almost uniformly* distributed in each partition.



Question:



How to be a proper node? <1>

To be a proper node (group),

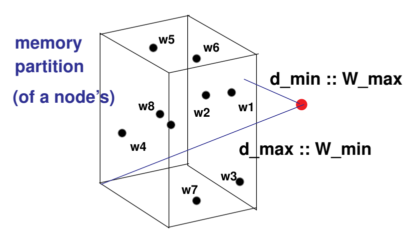
All the *individual weights* must be close enough.

Problem,

We *don't* calculate the individual weights, how can we know if they are close.

Solution,

W_{max} W_{min}

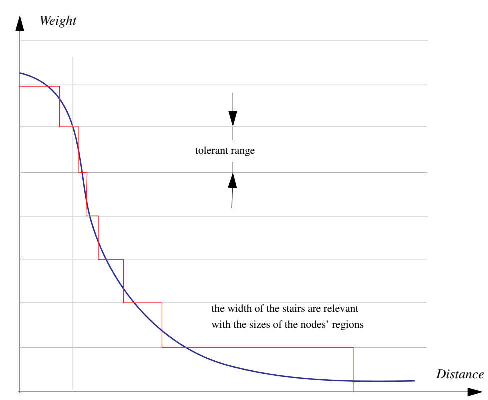


How to be a proper node? <2>

Criterion 1: $W_{max} - W_{min} < \tau$

Problem:

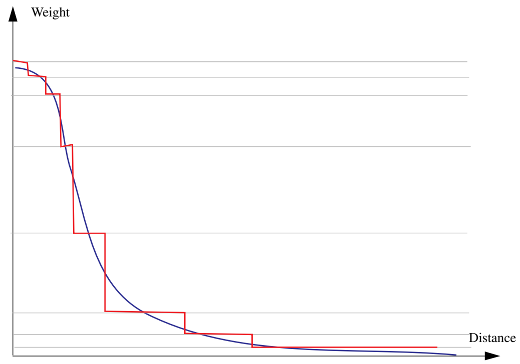
1. **The stair size at the *tail* is too big, inducing too large bias.**
2. **Not *normalized*, so that τ differs w.r.t. different problems.**



How to be a proper node? <3>

Criterion 2 :

$$(W_{max} - W_{min})\, M_i < \tau \sum W_i$$



An accumulated sum of W_i

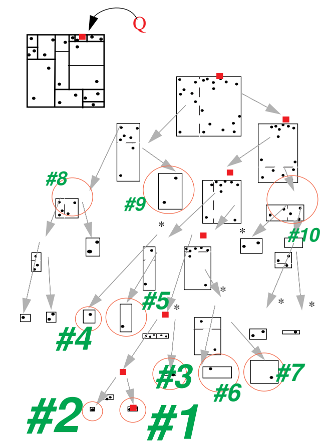
Criterion 2 (modified):

1. $(W_{max} - W_{min})\, M_i < \tau\, (\sum W_i$ ***So far)***

2. **Control the searching process,**

   start from **those nodes** ***close*** **to the query,**

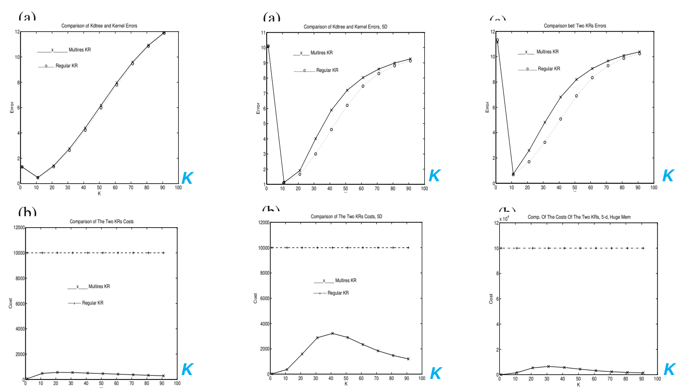
   then those nodes far away.



Experiments and Analysis <1>

Almost same accuracy

Much lower cost



2-d 10,000 datapnts

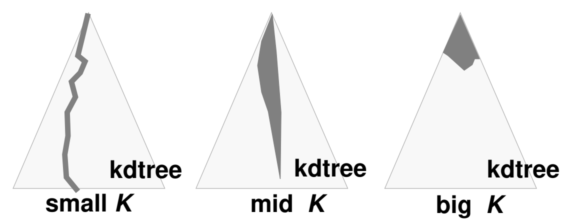
5-d 10,000 datapnts

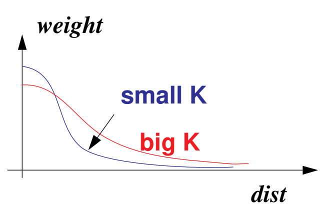
5-d 100,000 datapnts

Illustration of *K* spectrum

Small K: **peaky weighting function, like** *nearest neighbor.*

Big K: **flat weighting function, like** *average of all datapnts' outputs.*





Experiments and Analysis <2>

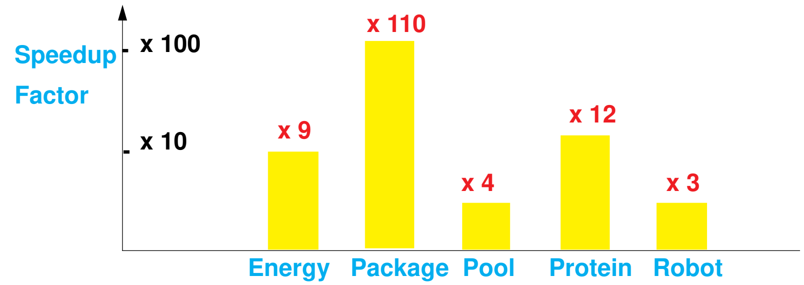
	9-d uniform	9-d inputs on 2d manifold
Regular KR cost	10,000	10,000
Multires KR cost	3,100	430
Regular KR mean testset error	13.07	1.06
Multires KR mean testset error	13.08	1.15

Table 1: Manufactured case: the input is embedded in a lower-dimensional manifold.

Experiments and Analysis <3>

Domain	Memory Size	Dimensionality	KR Cost	MRes Avg Cost	KR Error	MRes Error
Energy	2,144	5-d	2,144	232.9	1.687	1.690
Package	32,000	3-d	32,000	289.0	6.07	6.09
Pool	213	3-d	213	50.7	2.125	2.123
Protein	4,664	3-d	4,664	383.8	1.036	1.106
Robot	871	14-d	871	225.0	6.354	6.976

Table 1: Five real world cases.



Summary and Discussion

Advantages of Kernel Regression

1. *Larger memory, less bias.*
2. *No training phase.*
3. *Flexible from global to local.*

Disadvantages of Kernel Regression

1. *Mainly:*

   Expensively cost.
2. *Curse of dimentionality.*

Multi-resolution Kernel Regression

1. *Maintain the advantages.*
2. *Reduce the computational cost.*