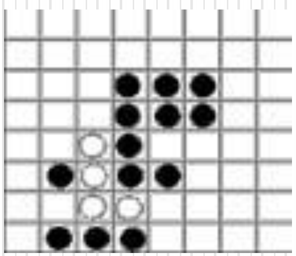
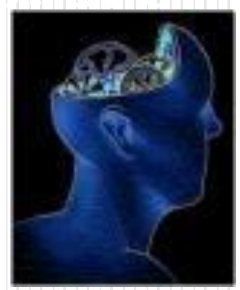
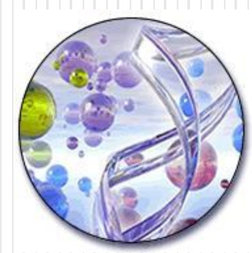
Welcome to

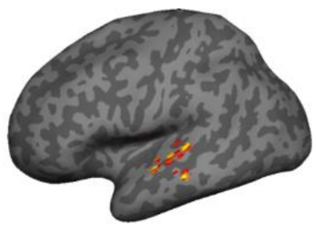
Introduction to Machine Learning!



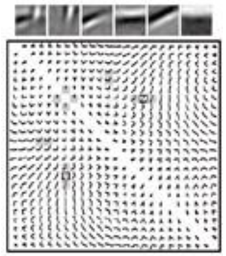




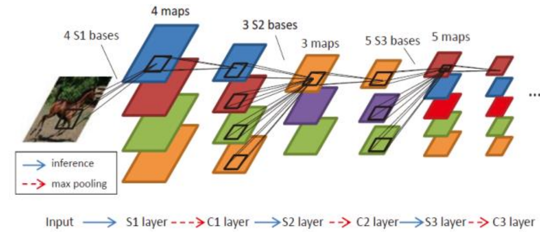
My research



Zhang et al., 2016

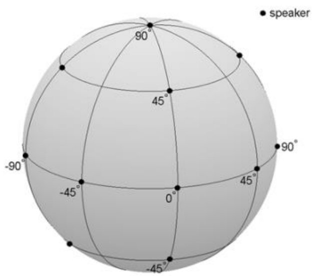


Hu et al., 2012, 2014

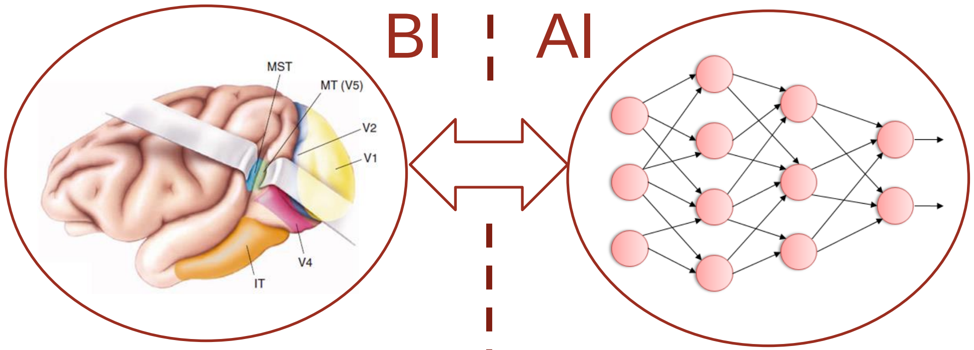


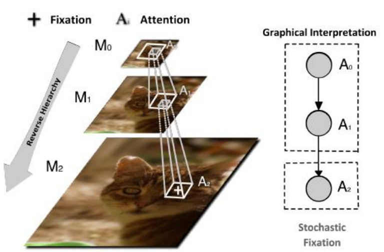
Hu et al., 2014

Liang et al., 2013;
Wu et al., 2013;
Li et al., 2015

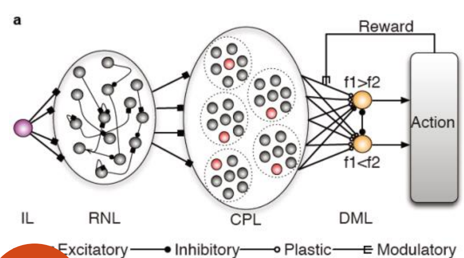


Zhang et al., 2015

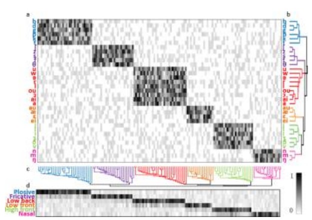




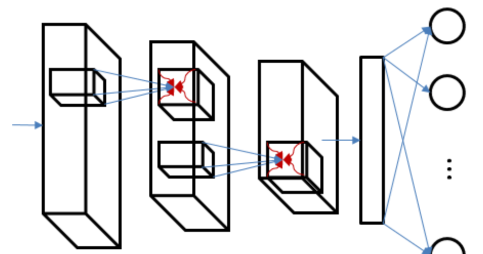
Shi et al., 2014



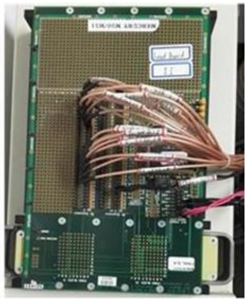
Cheng et al., 2015



Zhang et al., in preparation

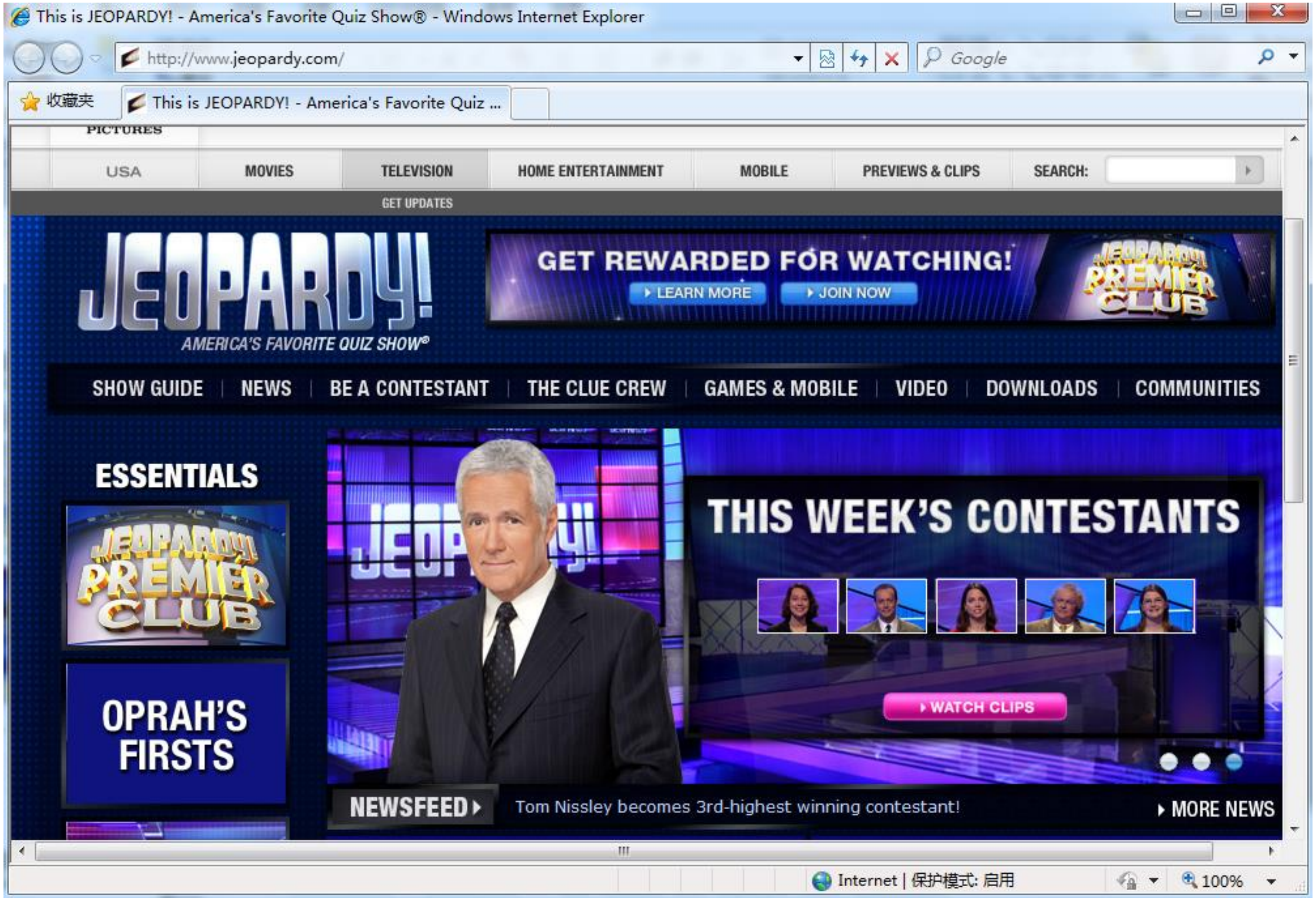


Liang et al., 2015a; 2015b



In progress

Coffee Time



Jeopardy: An American TV show

Requires the players to suss out the subtleties of language from jokes and puns to irony and anagrams

IBM Watson @ Jeopardy

- February 14, 15, and 16, 2011
  - *Jeopardy's* two biggest champions
  - Brad Rutter (right):
    - Won a whopping $3.25 million playing *Jeopardy*, the most cash ever awarded on the show.
    - He is a Johns Hopkins University dropout
  - Ken Jennings (left):
    - Holds the title for longest *Jeopardy* winning streak, with 74 consecutive wins in 2004.
    - He holds degrees in computer science and English, from Brigham Young University, and an international BA diploma from Seoul Foreign School.

IBM Watson won the Jeopardy

- http://domino.watson.ibm.com/library/cyberdig.nsf/papers/D12791EAA13BB952852575A1004A055C/$File/rc24789.pdf

  Towards the Open Advancement of Question Answering Systems

Final:

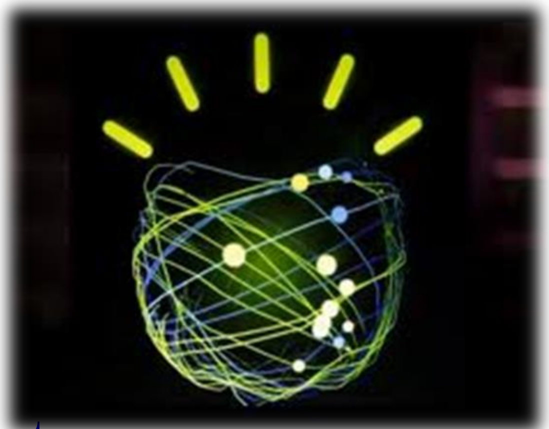
$77,147

to

$21,600 &

$24,000.

IBM Watson



- In development for 4 years
- Runs on 90 servers
- Does not connect to the Internet
- Search on a large scale knowledge base
- Trained with previous questions and games
  - With Jeopardy players: 77 (2009) + 55 (2010, winners)
  - Category: US Cities
  - *Its largest airport was named for a World War II hero; its second largest, for a World War II battle.*
  - *What is Chicago / Toronto ?*

Technical requirements

- Answers to questions on any topic
  - Science, geography, popular culture …
- Accuracy: not only an answer, but a confident right answer
- Speed: within 3 second or less

- Advanced linguistic understanding
  - Parser complex sentences, recognize and understand jokes, metaphors, puns and riddles
- Real time analysis of questions
- Learn from mistakes
- Be prepared to handle the unexpected …

Techniques involved -- DeepQA

- A massively parallel probabilistic evidence-based architecture for answering questions
  - Non-database approach
  - Deep text analytics
    - NLP and statistical NLP
  - Machine learning
    - Formulating parallel hypotheses with confidence score
    - Voting, Question interpretation…
  - Search
  - Risk assessment
  - Hadoop and UIMA

Topic 7. Instance-based learning

Xiaolin Hu

xlhu@tsinghua.edu.cn

Updated on Mar 31, 2017

Motivation

- Previous learning approaches
  - Make an assumption to the model
    - LSE, Decision Tree, MAP, MLE, Naïve Bayes, MM, HMM…
  - Find the optimal parameter

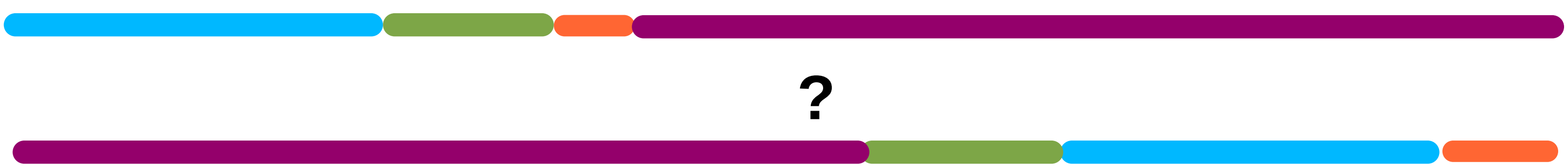
Is there a learning approach that is NOT "model assumption + parameter estimation" ?

Motivation

"Find the 10 most similar images to this image"

?

"Find all closest matching gene segments between two genomes"



Some Vocabulary

- Parametric (参数化) vs. Non-parametric(非参数化)
  - Parametric:
    - A particular functional form is assumed
    - Advantage of simplicity – easy to estimate and interpret
    - May have high bias because the real data may not obey the assumed functional form.
  - Non-parametric:
    - Distribution or density estimate is data-driven
    - Relatively few assumptions are made a priori about the functional form.

Instance-based Learning

- No model is built – Just store all training examples
- Any processing is delayed until a new instance must be classified.

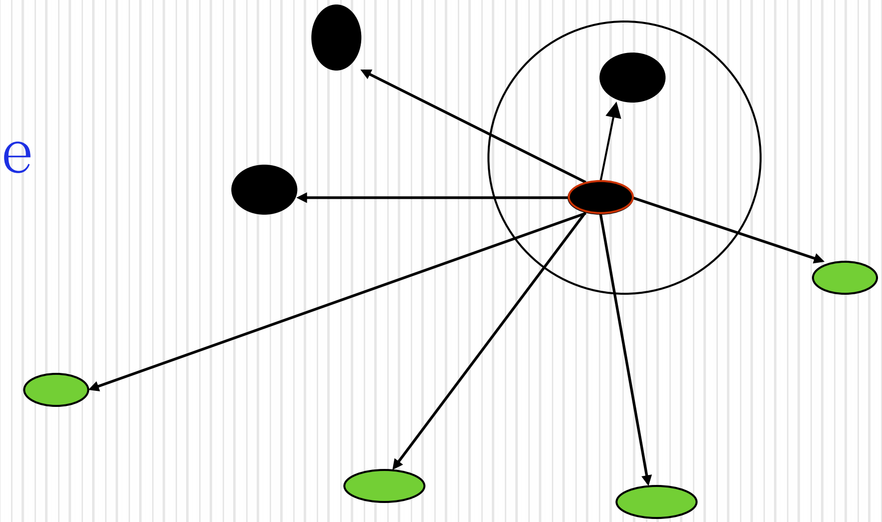
Non-parametric method

Outline

- Nearest Neighbor
- K-Nearest Neighbor
- KNN Discussion
- Distance-Weighted KNN

1. Nearest Neighbor

Similarity ←→ distance



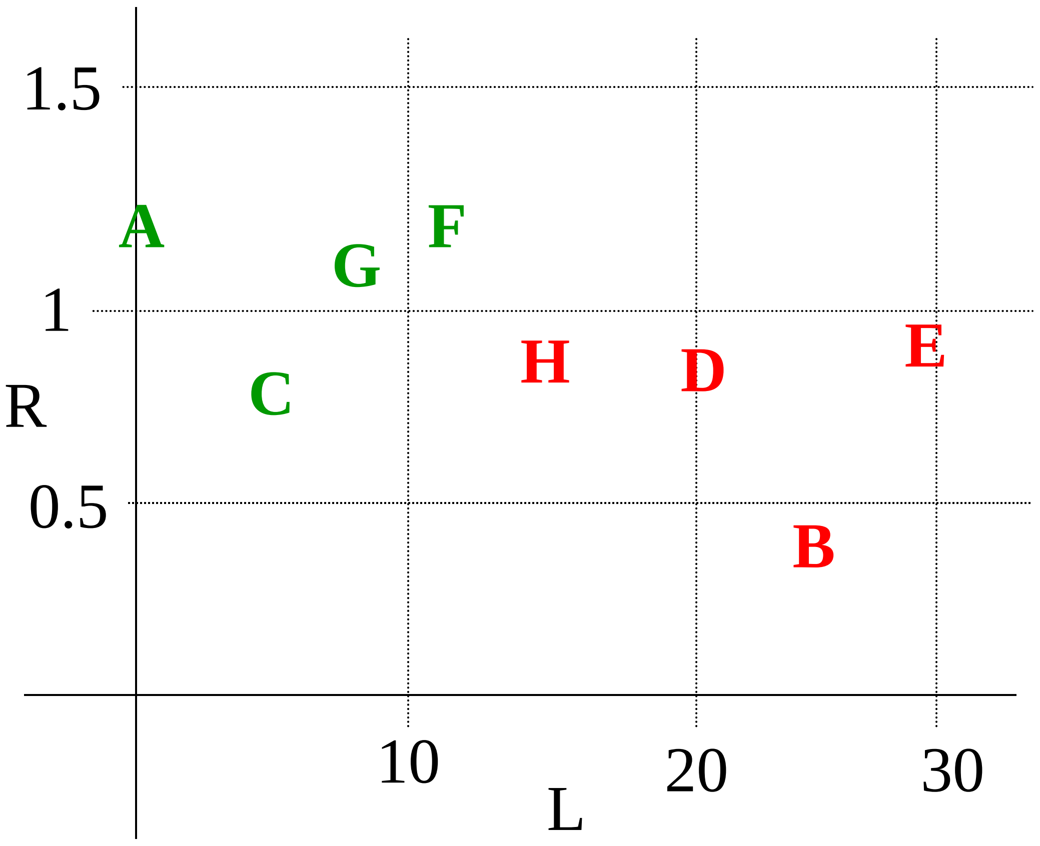
Nearest Neighbor Example

- Credit Rating
  - Classifier
    - Good / Poor
  - Features:
    - L = # late payments/yr;
    - R = Income/Expenses

name	L	R	G/P
A	0	1.2	G
B	25	0.4	P
C	5	0.7	G
D	20	0.8	P
E	30	0.85	P
F	11	1.2	G
G	7	1.15	G
H	15	0.8	P

Nearest Neighbor Example

name	L	R	G/P
A	0	1.2	G
B	25	0.4	P
C	5	0.7	G
D	20	0.8	P
E	30	0.85	P
F	11	1.2	G
G	7	1.15	G
H	15	0.8	P



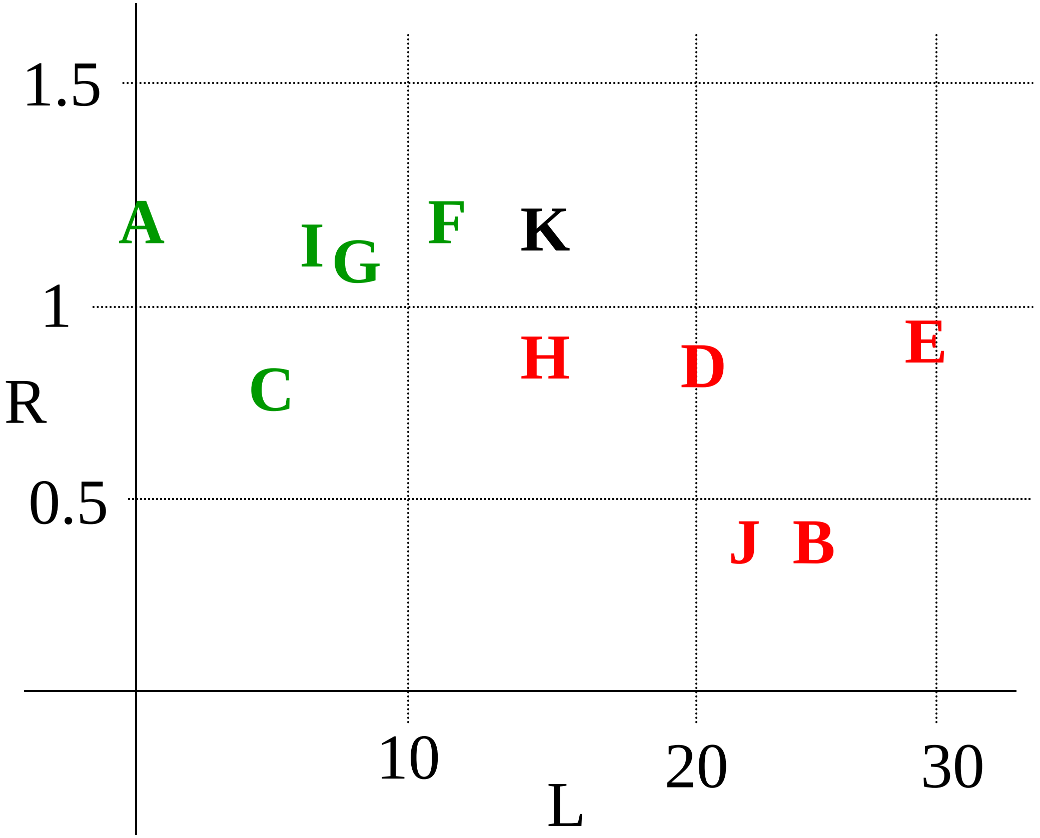
Nearest Neighbor Example (Cont.)

name	L	R	G/P
I	6	1.15	?
J	22	0.45	?
K	15	1.2	?

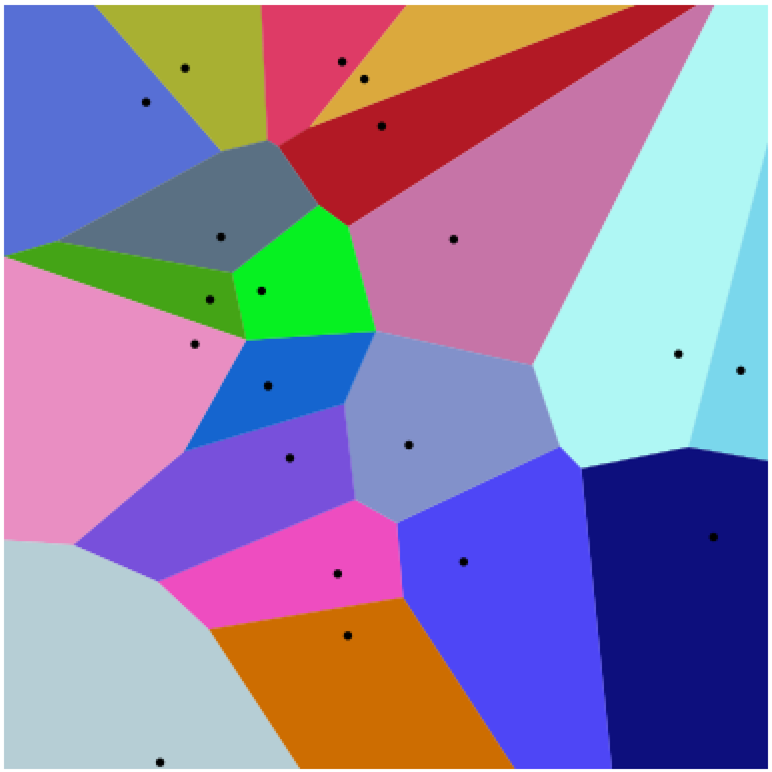
Distance Measure:

- Scaled distance

$$\sqrt{(L_1 - L_2)^2 + 100(R_1 - R_2)^2}$$



Nearest neighbor is related to Voronoi diagram



Euclidean distance

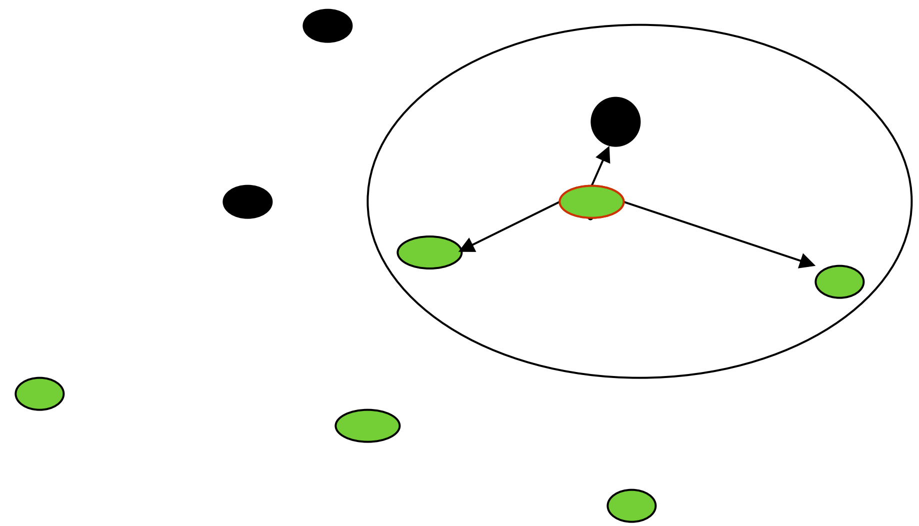
- The points p_k separate the space into adjacent cells R_k
- R_k consists of every point whose distance to p_k is less than or equal to its distance to any other p_k

From wikipedia

The problem

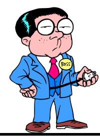
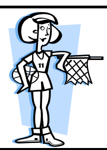
- ***What if the nearest neighbor is noise?***
  - Solution
    - Use more than one neighbors
    - Voting among neighbors

- **k-Nearest Neighbor**



2. K-Nearest Neighbor (KNN)

KNN: example (3-NN)

Customer	Age	Income (K)	#cards	Res	Distance from David
John 	35	35	3	No	sqrt $[(35-37)^2+(35-50)^2+(3-2)^2]=15.16$
Rachel 	22	50	2	Yes	sqrt $[(22-37)^2+(50-50)^2+(2-2)^2]=15$
Hannah 	63	200	1	No	sqrt $[(63-37)^2+(200-50)^2+(1-2)^2]=152.23$
Tom 	59	170	1	No	sqrt $[(59-37)^2+(170-50)^2+(1-2)^2]=122$
Nellie 	25	40	4	Yes	sqrt $[(25-37)^2+(40-50)^2+(4-2)^2]=15.74$
David 	37	50	2	Yes	

KNN discussion

KNN overview

- Basic algorithm
- Discussion
  - More distance metrics

KNN discussion 1: distance metrics

- Minkowski or L_λ metric:

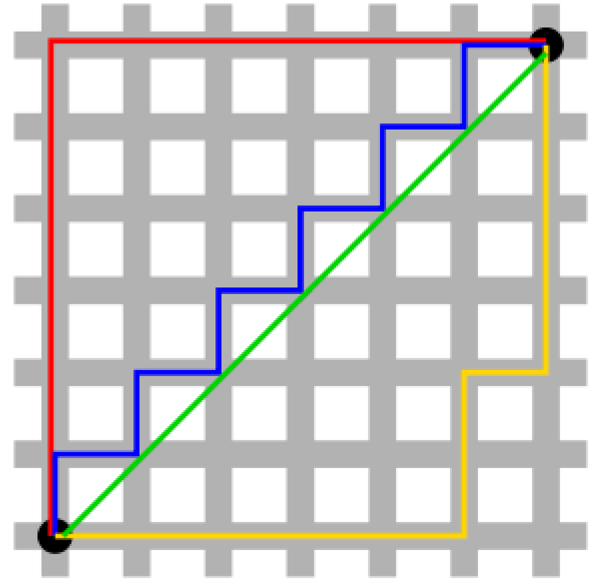
$$d(i,j) = \left(\sum_{k=1}^{p} | x_k(i) - x_k(j) |^{\lambda} \right)^{\frac{1}{\lambda}}$$

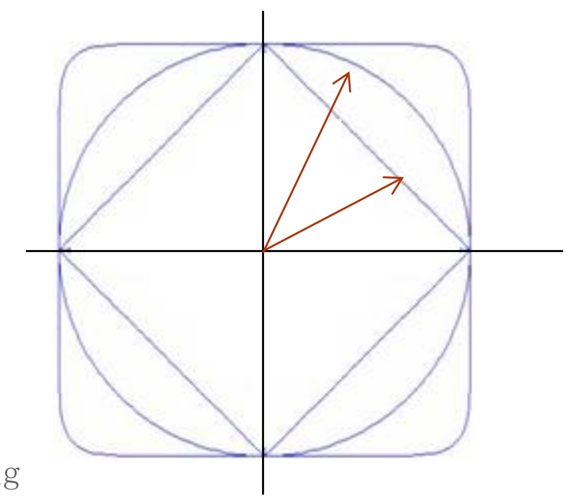
- Euclidean Distance $(\lambda = 2)$

$$d(i,j) = \sqrt{\sum_{k=1}^{p} (x_k(i) - x_k(j))^2}$$

- Manhattan dis., city block dis. $(\lambda = 1)$:

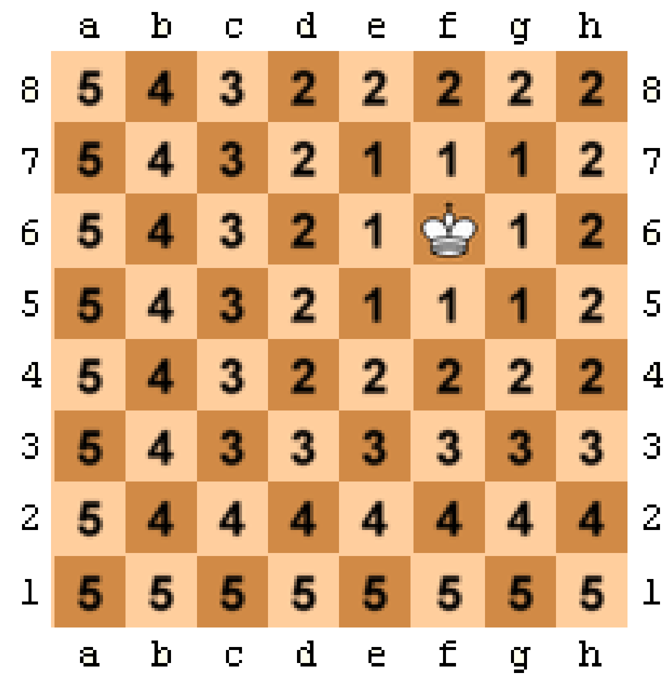
$$\mathrm{d(i,j)} = \sum_{\mathrm{k=1}}^{\mathrm{p}} \left| \mathrm{x_k(i) - x_k(j)} \right|$$





- Chebyshev distance, chessboard distance, L_∞

$$d(i,j) = \max_k \left| x_k(i) - x_k(j) \right|$$



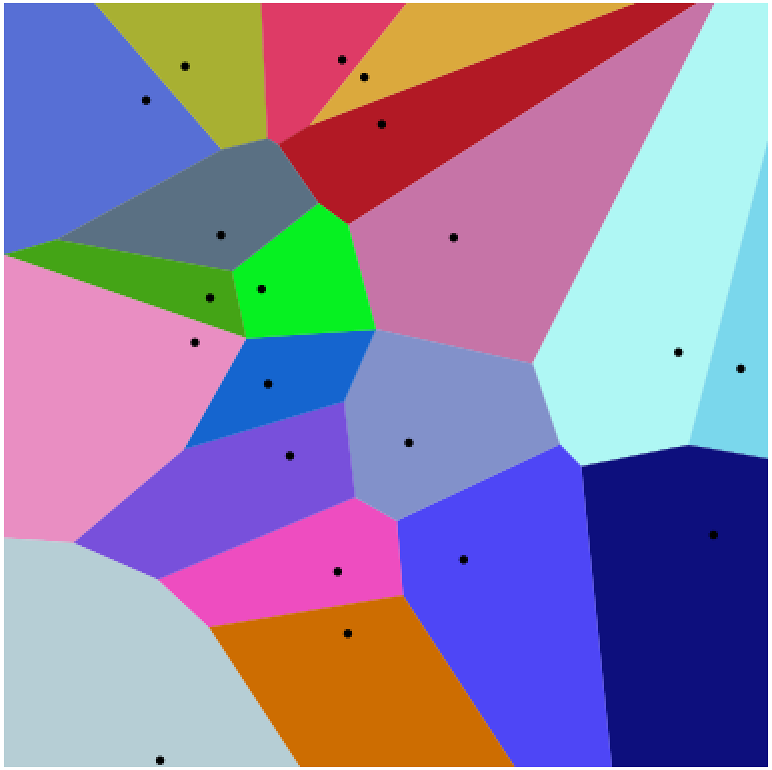
- Mean Censored Euclidean :

$$\sqrt{\sum_k (x_{ik} - x_{jk})^2 / n}$$

- Bray-Curtis $\sum_k \left| x_{ik} - x_{jk} \right| \Big/ \sum_k (x_{ik} + x_{jk})$

- Canberra $$\frac{\sum_k \left| x_{ik} - x_{jk} \right| \Big/ (x_{ik} + x_{jk})}{k}$$

Voronoi diagram with different distance metrics



Euclidean distance

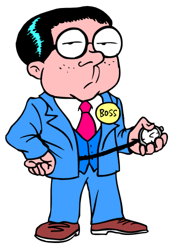
Manhattan distance

From wikipedia

KNN overview

- Basic algorithm
- Discussion
  - More distance metrics
  - Attributes

KNN discussion 2: attributes



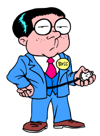
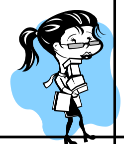
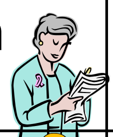
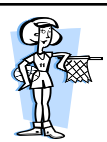
John:
Age=35
Income=95K
No. of credit cards=3

Rachel:
Age=41
Income=215K
No. of credit cards=2

$$Dis\ (John,\ Rachel)=sqrt\ [(35-45)^2 + (95,000-215,000)^2 + (3-2)^2]$$

- Distance between neighbors could be dominated by some attributes with relatively large numbers
  - e.g. Income
- Important to normalize some features
  - e.g., map numbers to [0-1]

KNN: attributes normalization

Customer	Age	Income (K)	#cards	Response
John 	35/63= 0.55	35/200= 0.175	3/4= 0.75	No
Rachel 	22/63= 0.34	50/200= 0.25	2/4= 0.5	Yes
Hannah 	63/63= 1	200/200= 1	1/4= 0.25	No
Tom 	59/63= 0.93	170/200= 0.85	1/4= 0.25	No
Nellie 	25/63= 0.39	40/200= 0.2	4/4= 1	Yes
David 	37/63= 0.58	50/200= 0.25	2/4= 0.5	Yes

KNN: weighted attributes

- The classification of an example is based on all the attributes
  - Independent of their relevance – Even the irrelevant attributes are used.
- Weight the contribution of each attribute based on its relevance, e.g.

$$d_{WE}(i, j) = \left(\sum_{k=1}^{p} w_k (x_k(i) - x_k(j))^2 \right)^{\frac{1}{2}}$$

- Scaling of dimensions in the distance space
  - $w_k = 0$ → eliminate the corresponding dimension (feature selection)

- Possible weighting method:
  - use mutual information *I*(*each attribute*, *the class*)
  - Determine automatically by cross validation

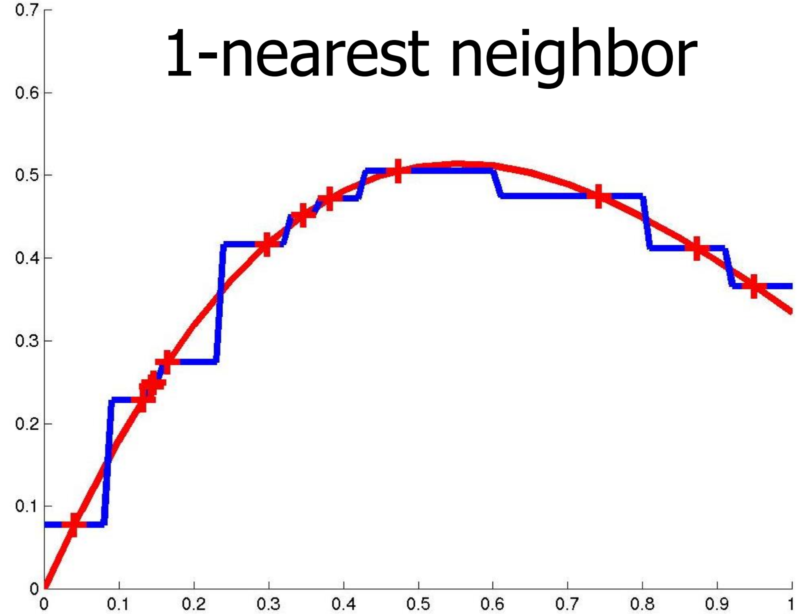
KNN overview

- Basic algorithm
- Discussion
  - More distance metrics
  - Attributes
    - Normalization, Weighting
  - Continuous valued target function

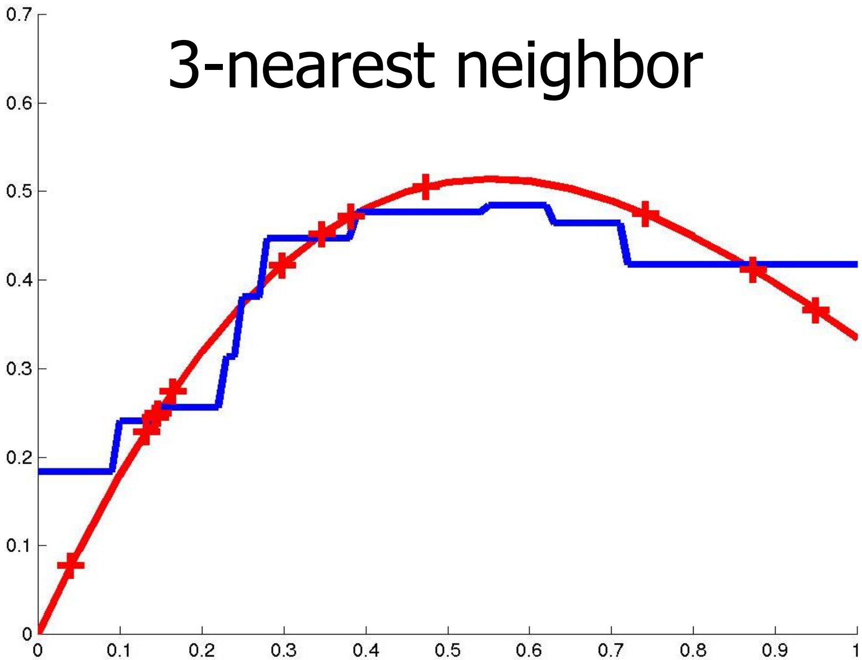
KNN discussion 3: continuous valued target function

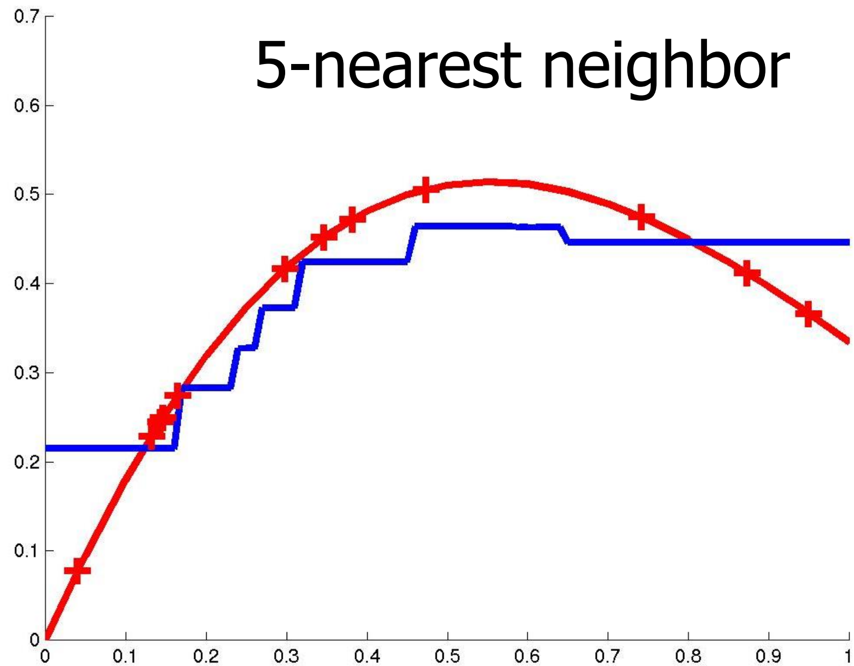
- Discrete output -- voting
- Continuous valued target function
  - Mean value of the k nearest training examples

Continuous valued target function



Red: known instances
Blue: estimation





KNN overview

- Basic algorithm
- Discussion
  - More distance metrics
  - Attributes
    - Normalization, Weighting
  - Continuous valued target function
  - Choose k

KNN discussion 4: choose k

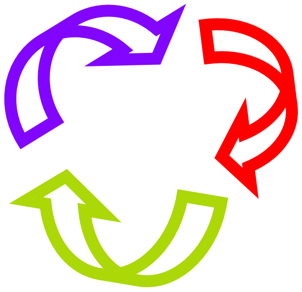
- In many cases k=3
- Depending on the number of training examples
  - Larger k doesn’t mean better performance
- Cross validation
  - Leave-one-out (Throw-one-out, Hold-one-out)
    - Each time: Take one sample to test, the others are all training examples
- KNN is stable
  - Small perturbation of training data does not change results significantly.

KNN overview

- Basic algorithm
- Discussion
  - More distance metrics
  - Attributes
    - Normalization, Weighting
  - Continuous valued target function
  - Choose k
  - Break ties

KNN discussion 5: break ties

- What if k=3, and each near neighbor belongs to a different class?
  - $P(w \mid X)=1/3$
  - or find a new neighbor (4th)
  - or choose the nearest one
  - or random select one
  - or ...



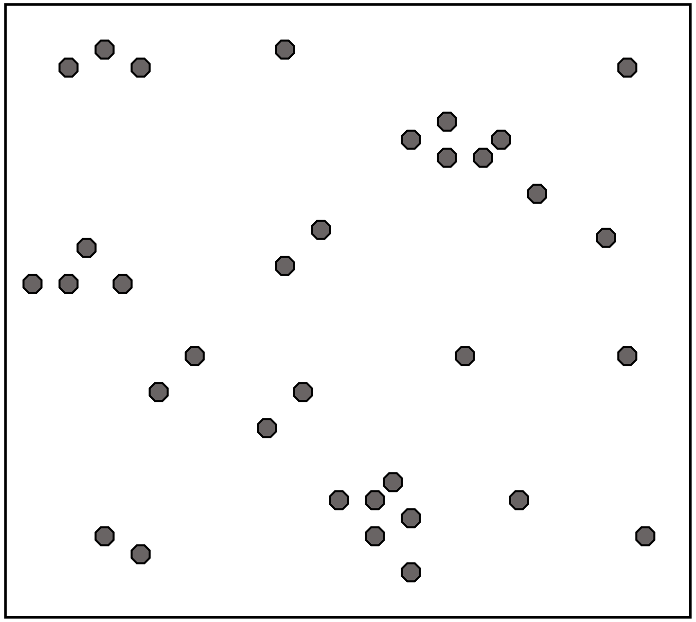
KNN overview

- Basic algorithm
- Discussion
  - More distance metrics
  - Attributes
    - Normalization, Weighting
  - Continuous valued target function
  - Choose k
  - Break ties
  - More on efficiency

KNN discussion 6: more on efficiency

- We can speed up the search for the nearest neighbor:
  - Examine nearby points first.
  - Ignore any points that are further than the nearest point found so far.
- Do this using a KD-tree:
  - KD-tree: k dimensional tree (the dimension of the points is k)
  - Tree-based data structure
  - Recursively partitions points into axis aligned boxes.

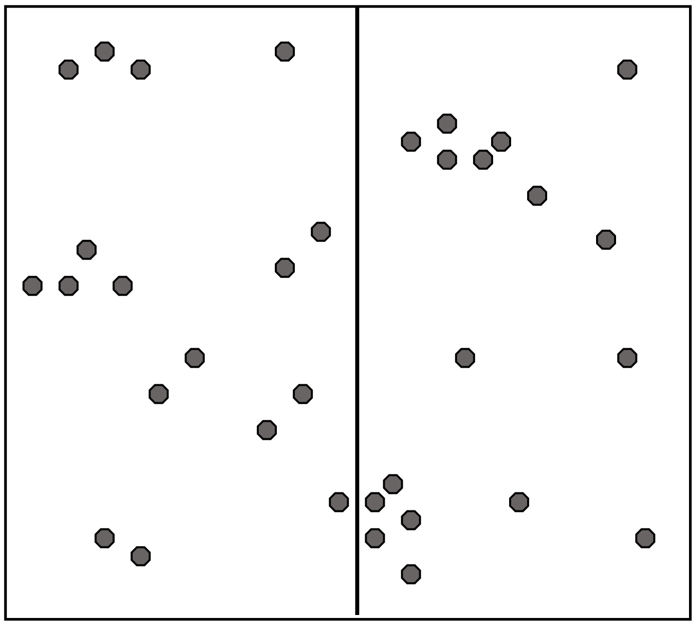
KD-Tree: Construction

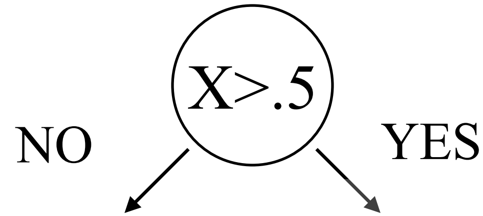


Pt	X	Y
1	0.00	0.00
2	1.00	4.31
3	0.13	2.85
…	…	…

We start with a list of points.

KD-Tree: (1) Construction



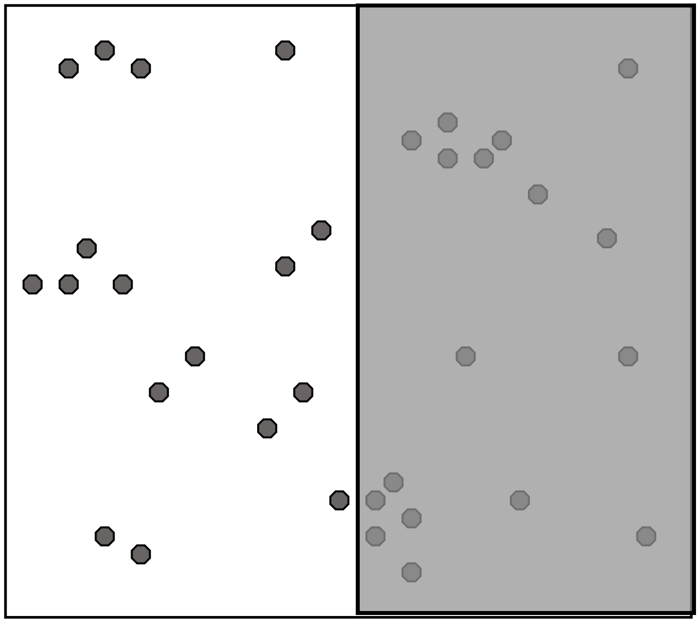


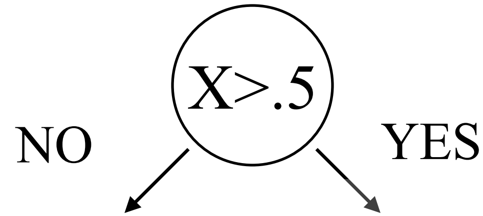
Pt	X	Y
1	0.00	0.00
3	0.13	2.85
…	…	…

Pt	X	Y
2	**1.00**	**4.31**
…	**…**	**…**

We can split the points into 2 groups by choosing a dimension X and value V and separating the points into X > V and X <= V.

KD-Tree: (1) Construction



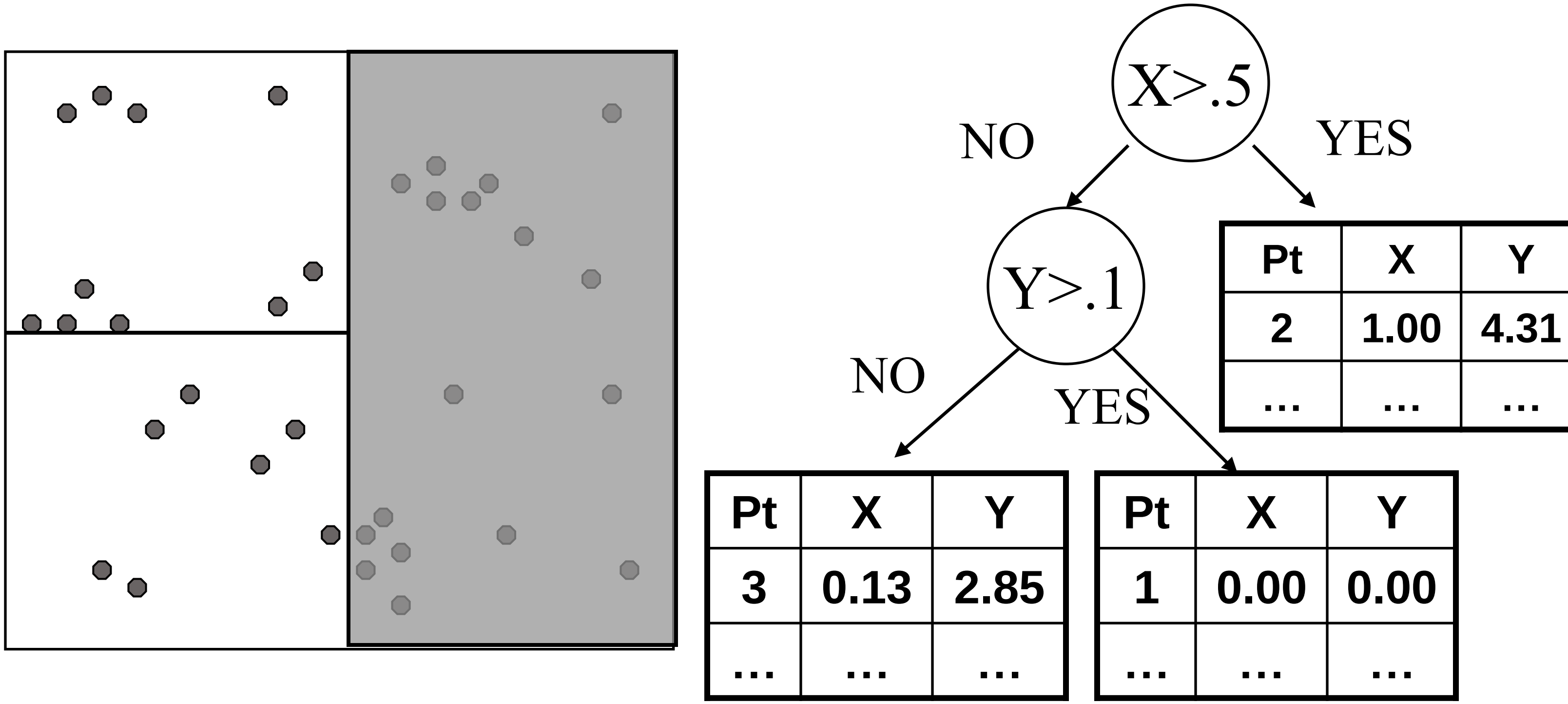


Pt	X	Y
1	0.00	0.00
3	0.13	2.85
…	…	…

Pt	X	Y
2	1.00	4.31
…	…	…

We can then consider each group separately and possibly split again (along same/different dimension).

KD-Tree: (1) Construction



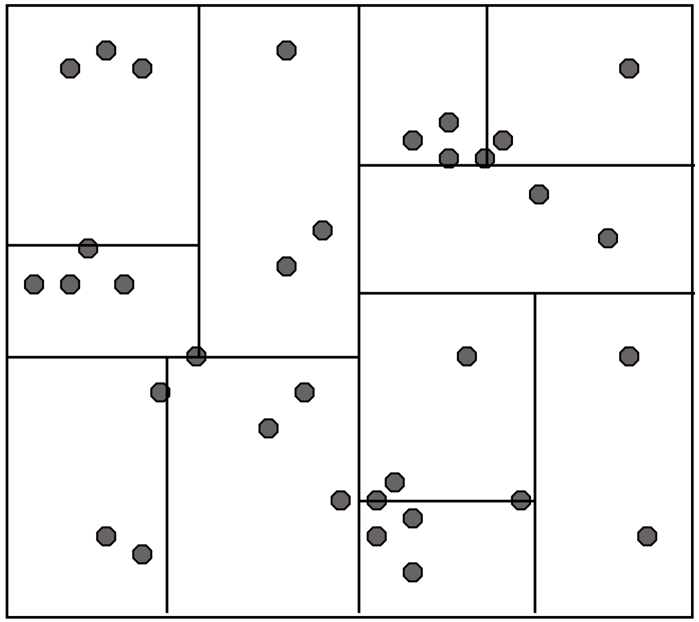
Pt	X	Y
2	1.00	4.31
…	…	…

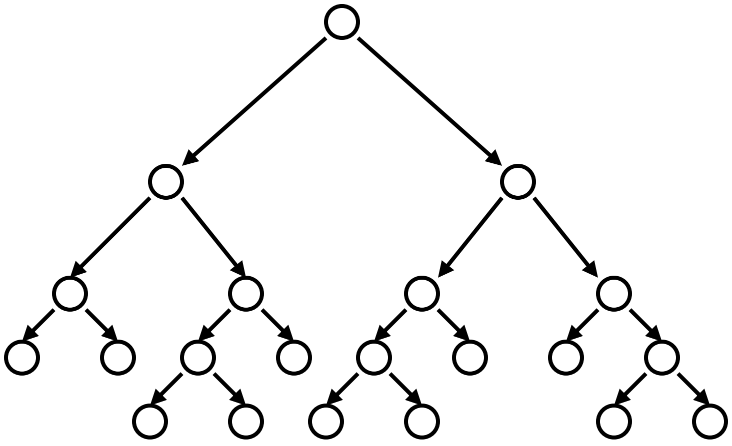
Pt	X	Y
3	0.13	2.85
…	…	…

Pt	X	Y
1	0.00	0.00
…	…	…

We can then consider each group separately and possibly split again (along same/different dimension).

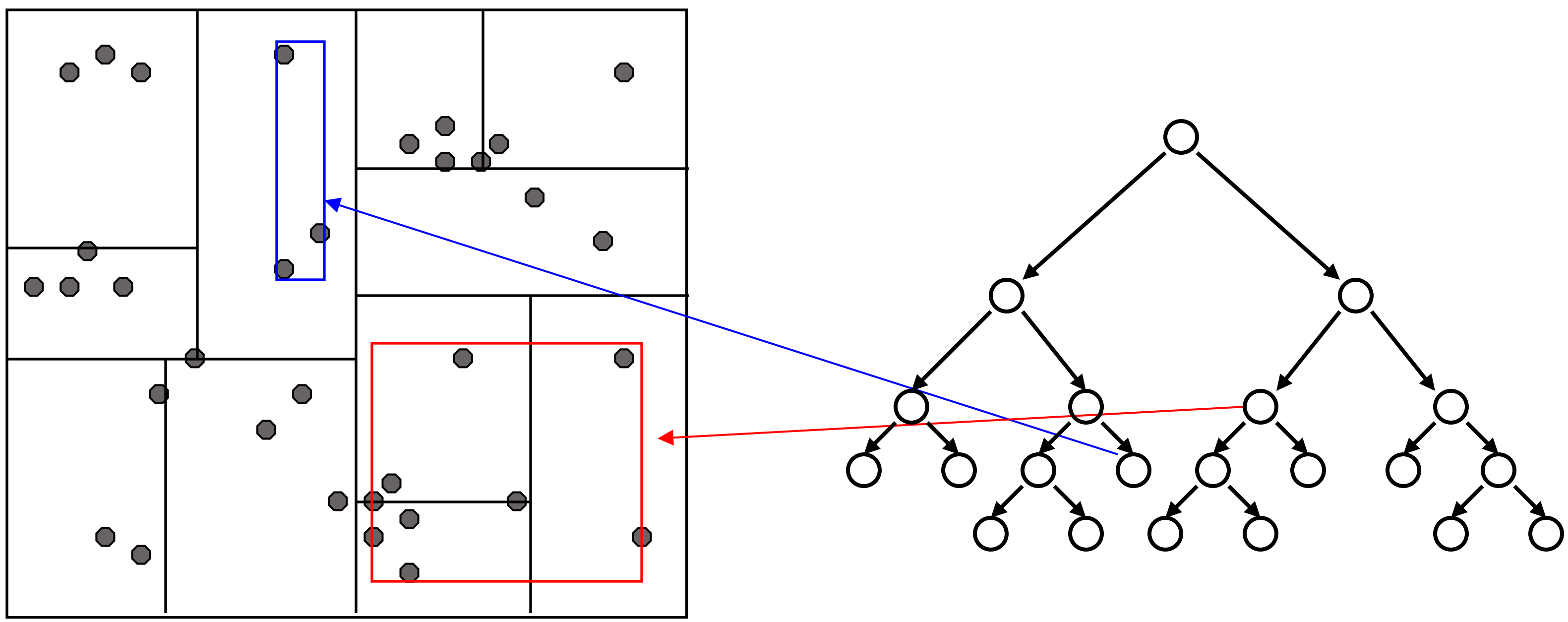
KD-Tree: (1) Construction





We can keep splitting the points in each set to create a tree structure. Each leaf node contains a list of points.

KD-Tree: (1) Construction



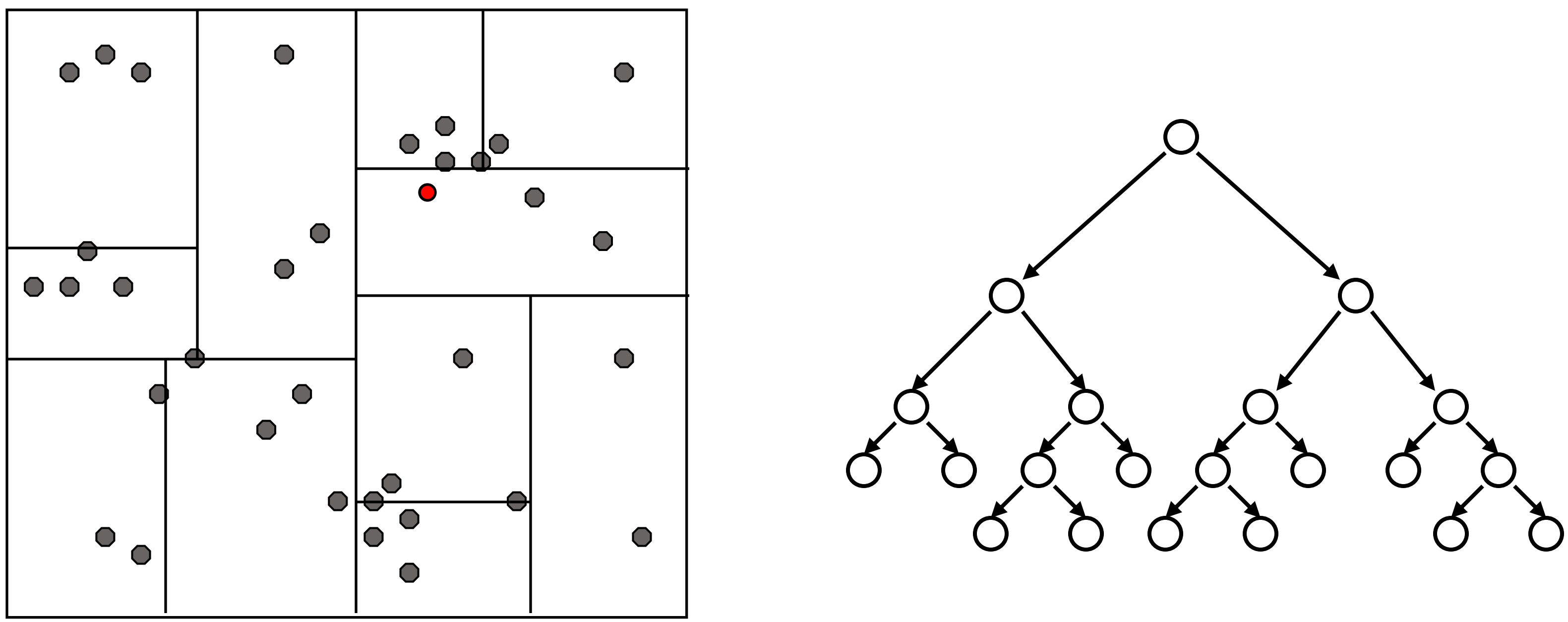
We will keep around one additional piece of information at each node: **The (tight) bounds of the points at or below this node**.

KD-Tree: (1) Construction

Use heuristics to make splitting decisions:

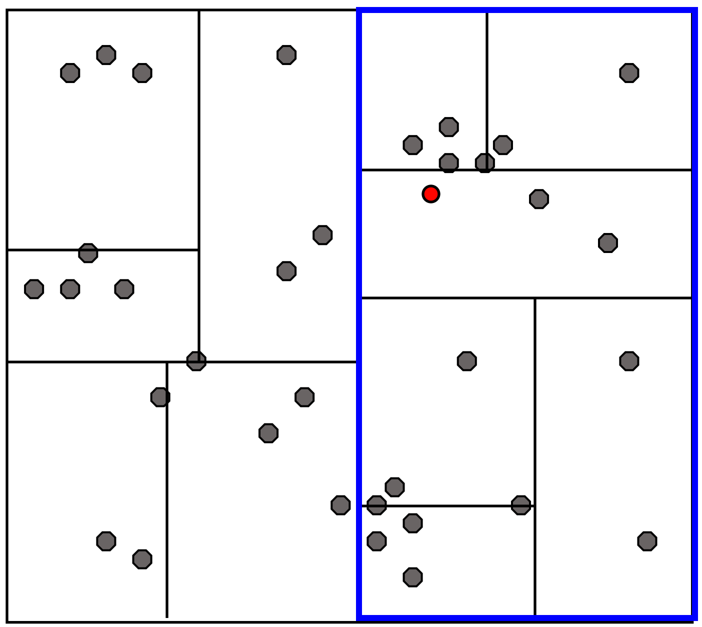
- Which dimension do we split along?
  - Widest
- Which value do we split at?
  - Median of value of that split dimension for the points.
- When do we stop?
  - When there are fewer then m points left, OR
  - The box has hit some minimum width.

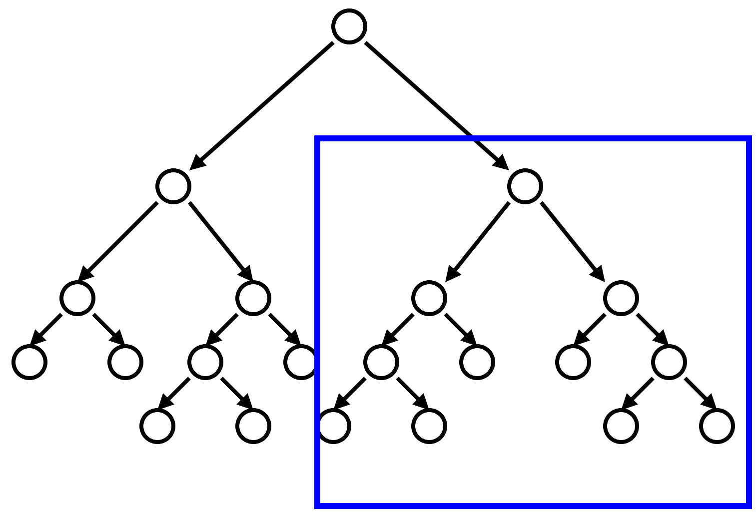
KD-Tree: (2) Query



We traverse the tree looking for the nearest neighbor of the query point.

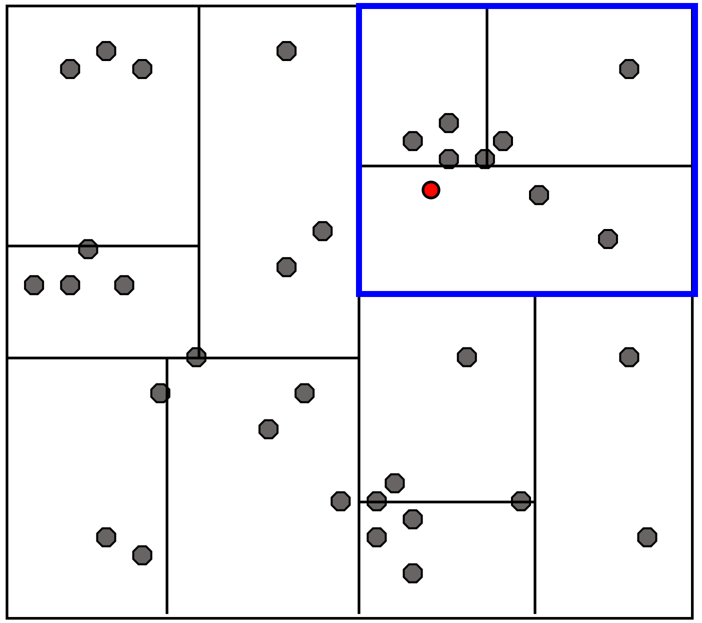
KD-Tree: (2) Query

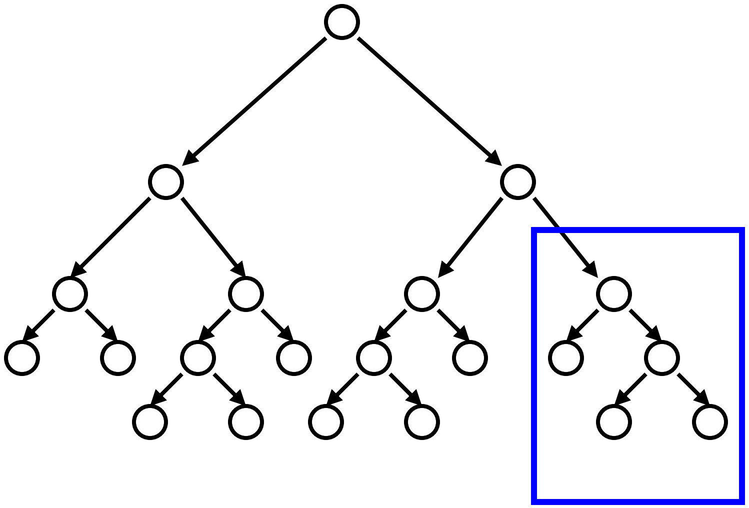




Examine nearby points first: Explore the branch of the tree that is closest to the query point first.

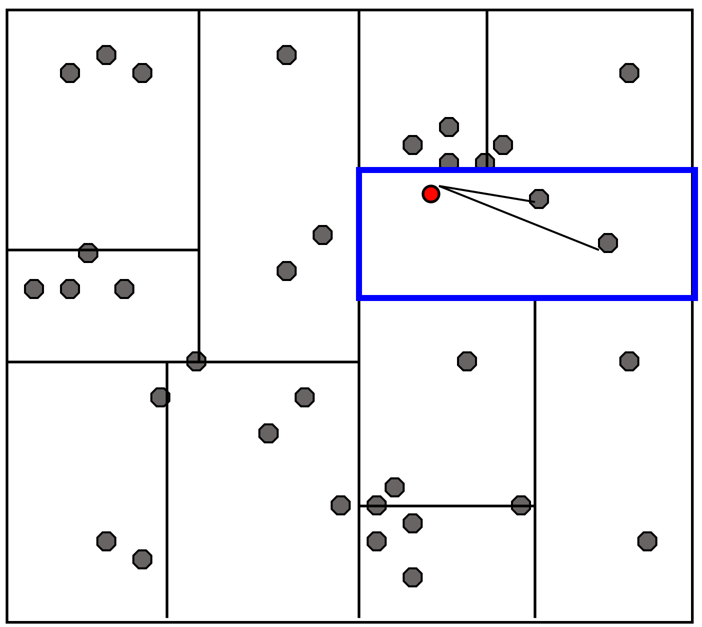
KD-Tree: (2) Query

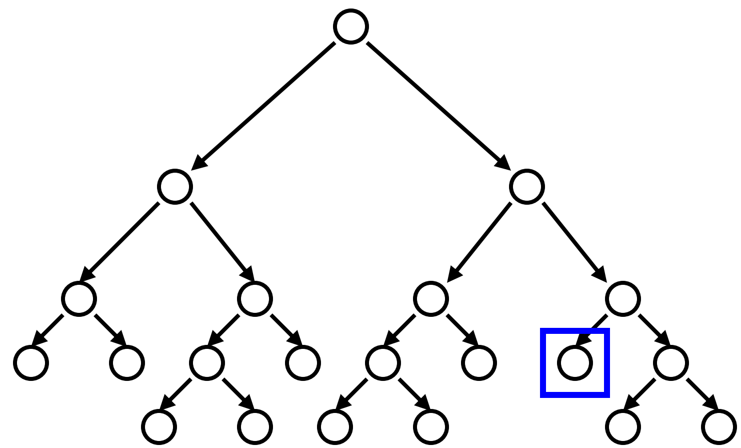




Examine nearby points first: Explore the branch of the tree that is closest to the query point first.

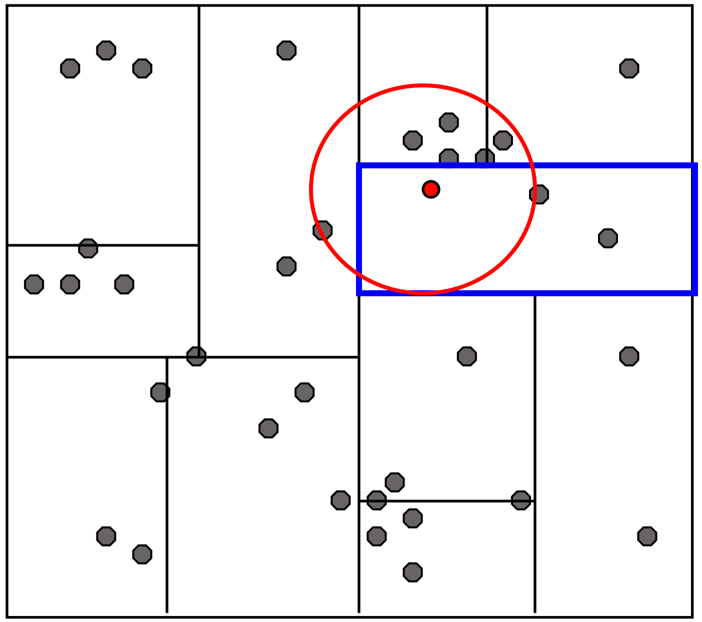
KD-Tree: (2) Query

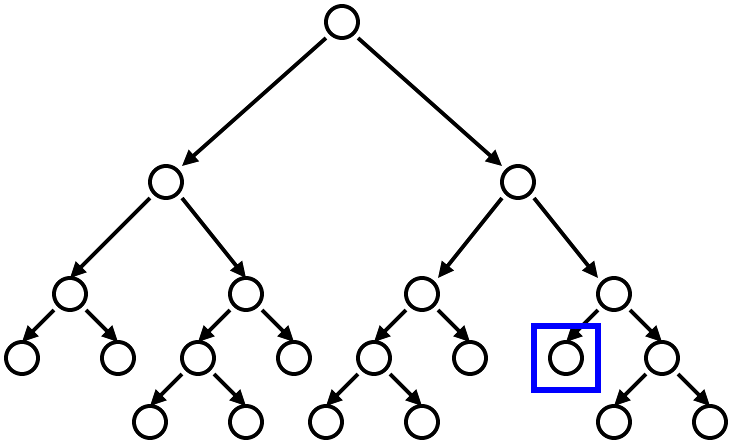




When we reach a leaf node: compute the distance to each point in the node.

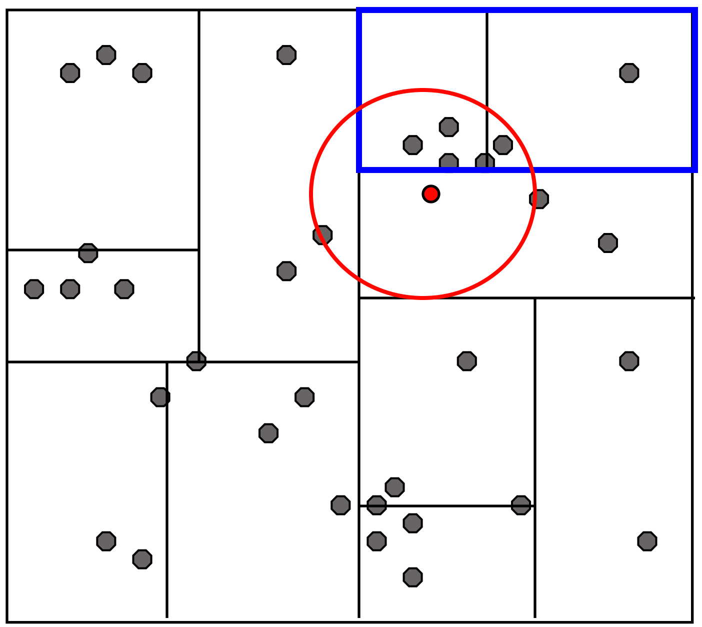
KD-Tree: (2) Query

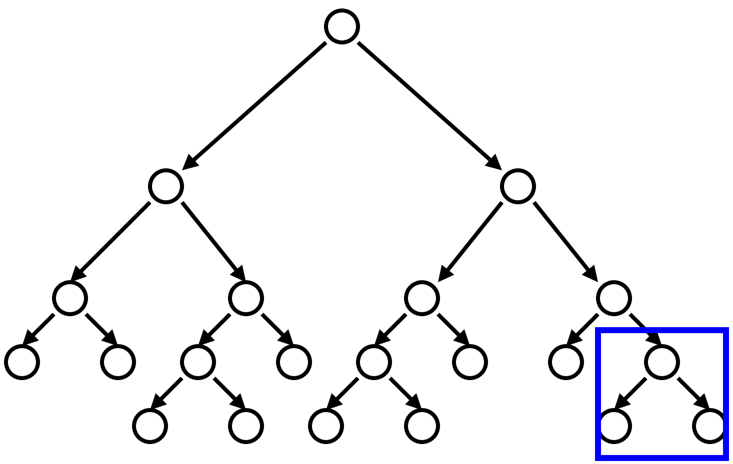




When we reach a leaf node: compute the distance to each point in the node.

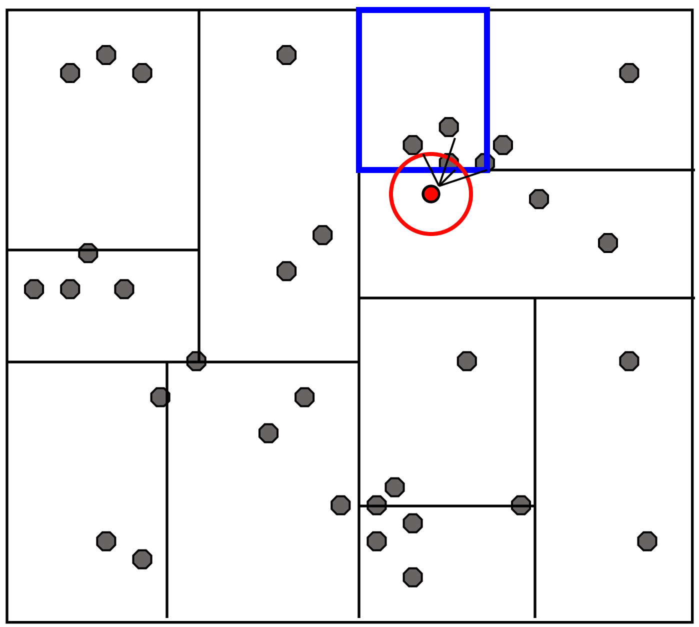
KD-Tree: (2) Query

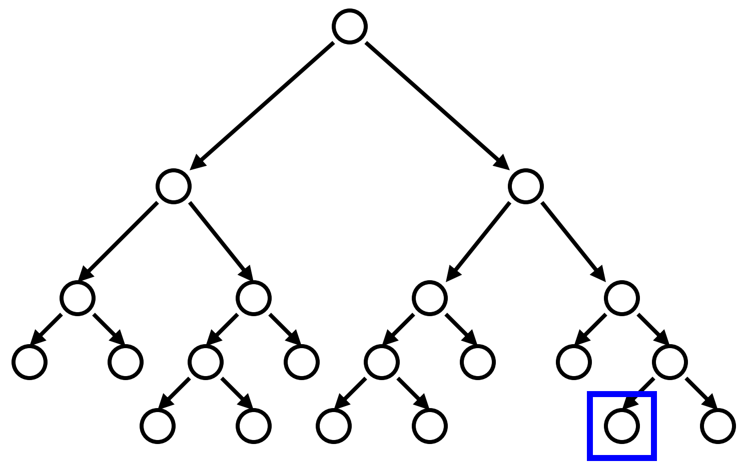




Then we can backtrack and try the other branch at each node visited.

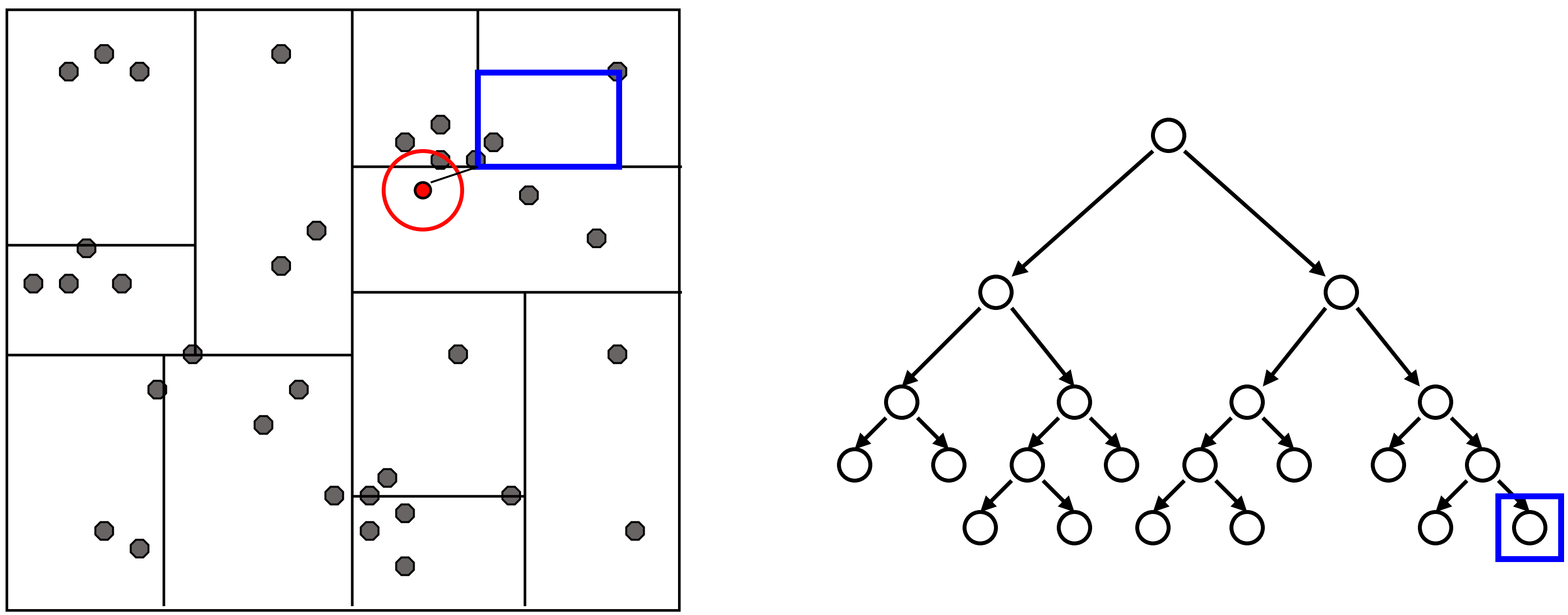
KD-Tree: (2) Query





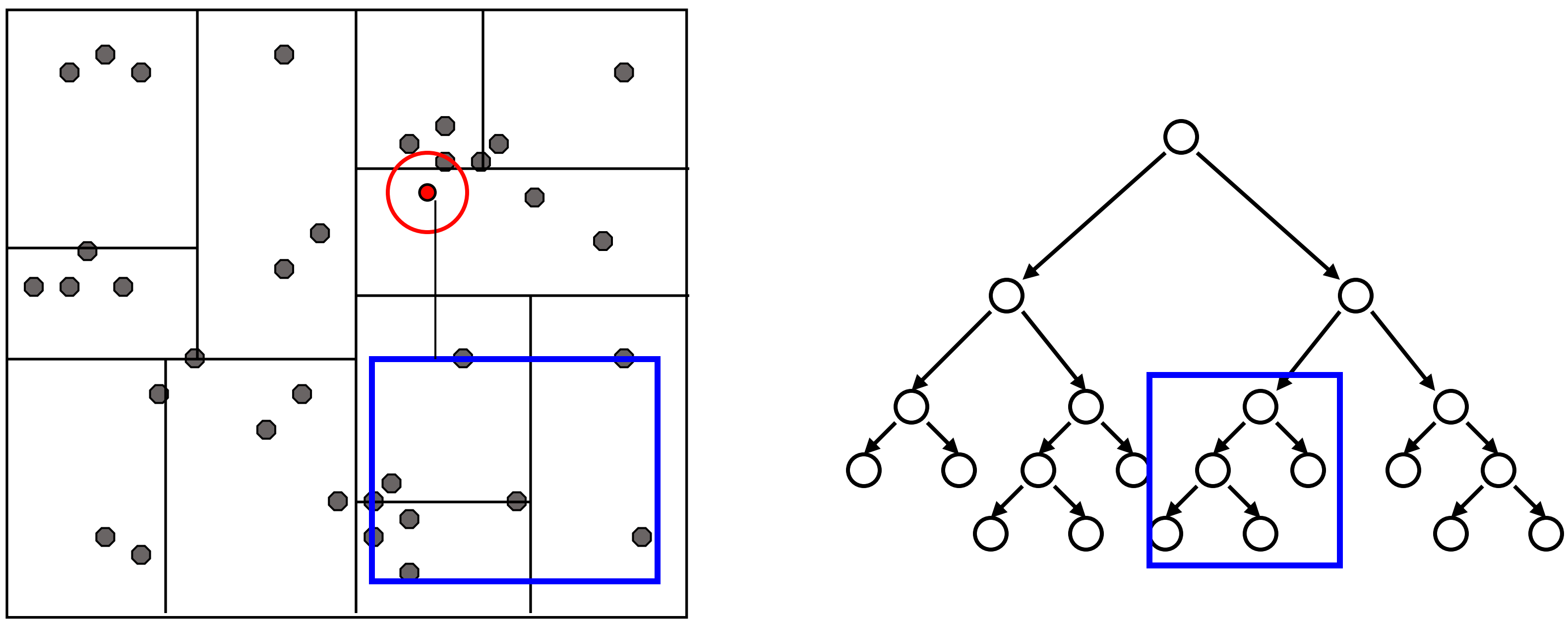
Each time a new closest node is found, we can update the distance bounds.

KD-Tree: (2) Query



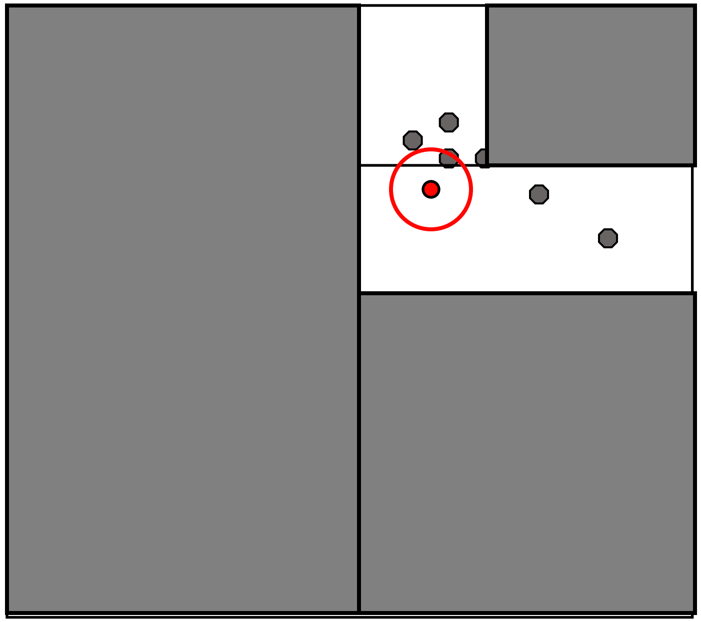
Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

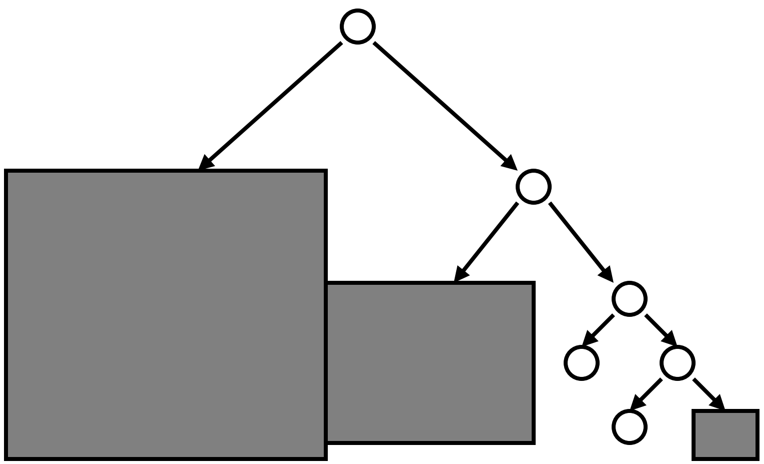
KD-Tree: (2) Query



Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

KD-Tree: (2) Query





Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

KNN overview

- Basic algorithm
- Discussion
  - More distance metrics
  - Attributes
    - Normalization, Weighting
  - Continuous valued target function
  - Choose k
  - Break ties
  - More on efficiency - k-Dtree
- Strength and weakness

KNN overview: strength

- Conceptually simple, yet able to form comp
  (and complex target functions)
  - e.g. image classification
- Robust to noisy data by averaging k-nearest neighbors
- Comprehensible
  - Easy to explain prediction (near neighbor)
- Information present in the training examples is never lost
  - Because the examples themselves are stored explicitly.
- Simple to implement, stable, no parameters (except for k), easy to run leave-one-out test.

KNN overview: weakness

- Memory cost
  - Need a lot of space to store all examples
  - Generally, it requires distances between all pairs of points O(n^2)
    - K-DTrees O($n \log n$)
- CPU cost
  - Takes more time to classify a new example (therefore in offline application)
- It is difficult to determine an appropriate distance function
  - Especially when examples are represented as complex symbolic expressions.
- Irrelevant features have a negative impact on the distance metric.

The next problem

- Recall: Use more than one neighbors to be robust to noise data

Do these neighbors have same contributions?

- Solution
  - Weighting the data
  - Give larger weights to neighbors closer to the query

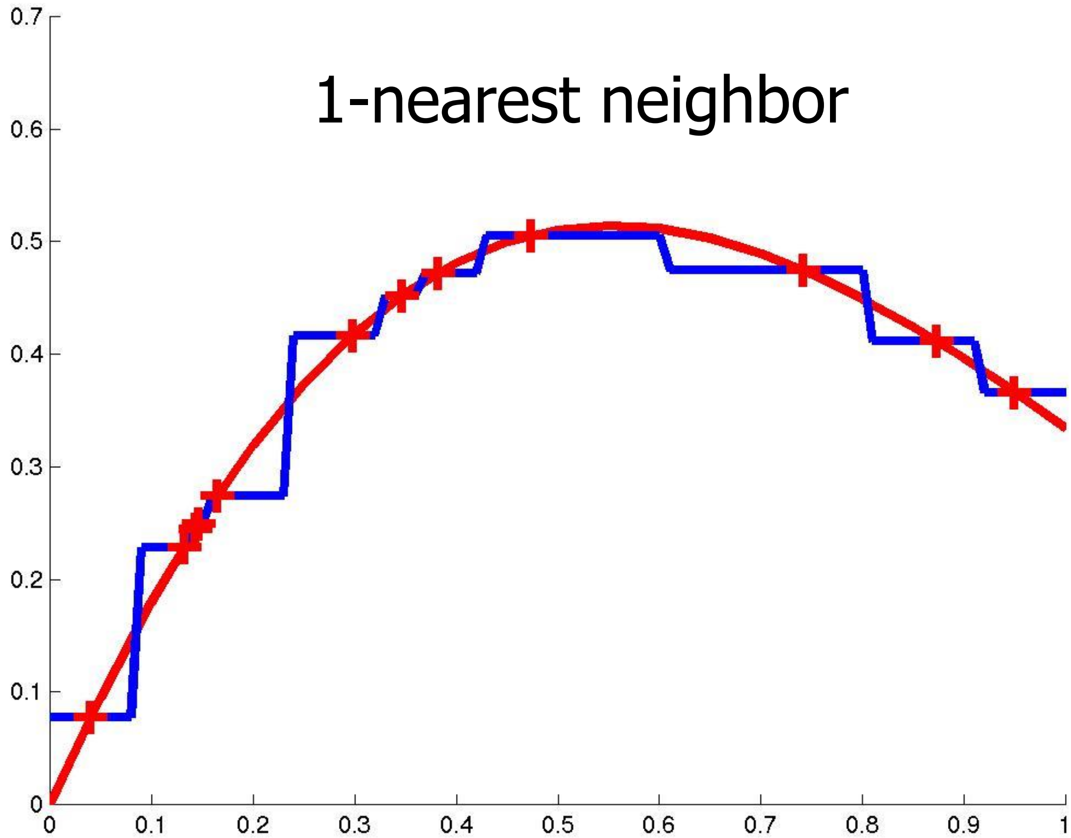
Distance-weighted Nearest Neighbor

3. Distance weighted KNN

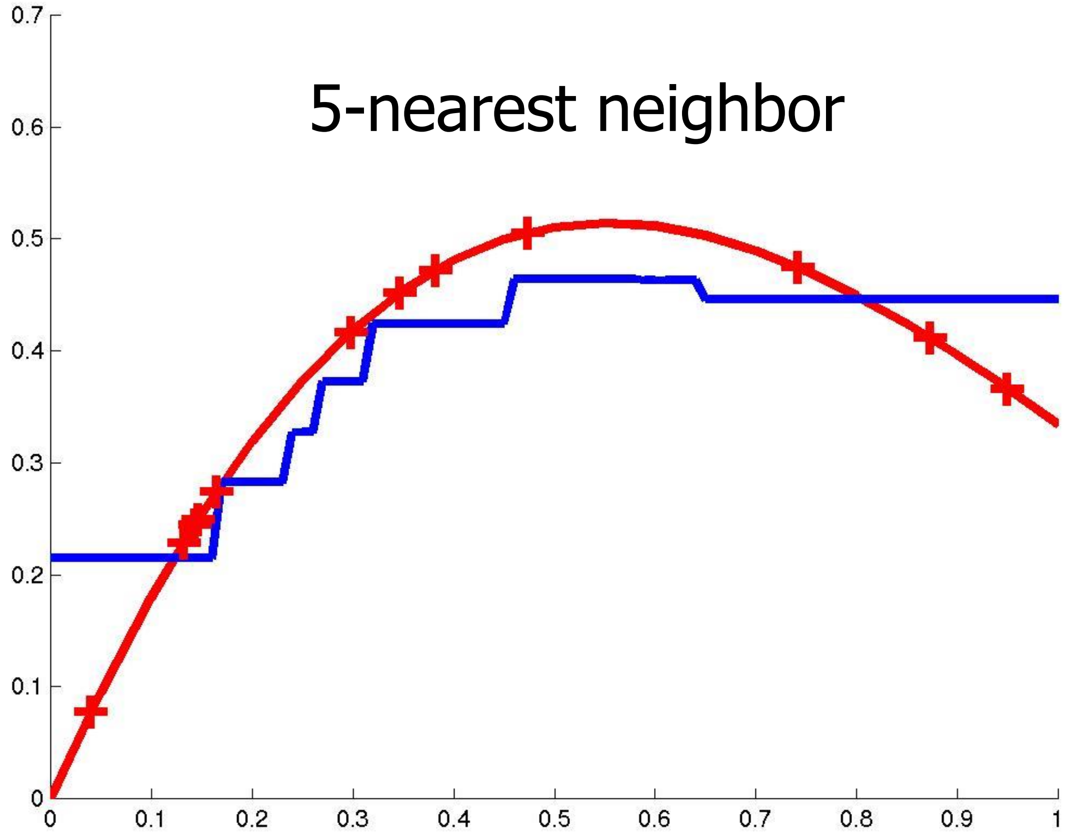
Distance Weighted KNN

- A weighting function
  - $w_i = K(d(x_i, x_q))$
  - $d(x_i, x_q)$ is the distance between query and x_i
  - K - kernel function that determines the weight for each point
- Output:
  - weighted average: predict = $\Sigma w_i y_i / \Sigma w_i$
- Kernel function $K(d(x_i, x_q))$
  - $1/d^2$, e^{-d}, $1/(1+d)$, …
  - Should vary inversely with the distance d

Recall

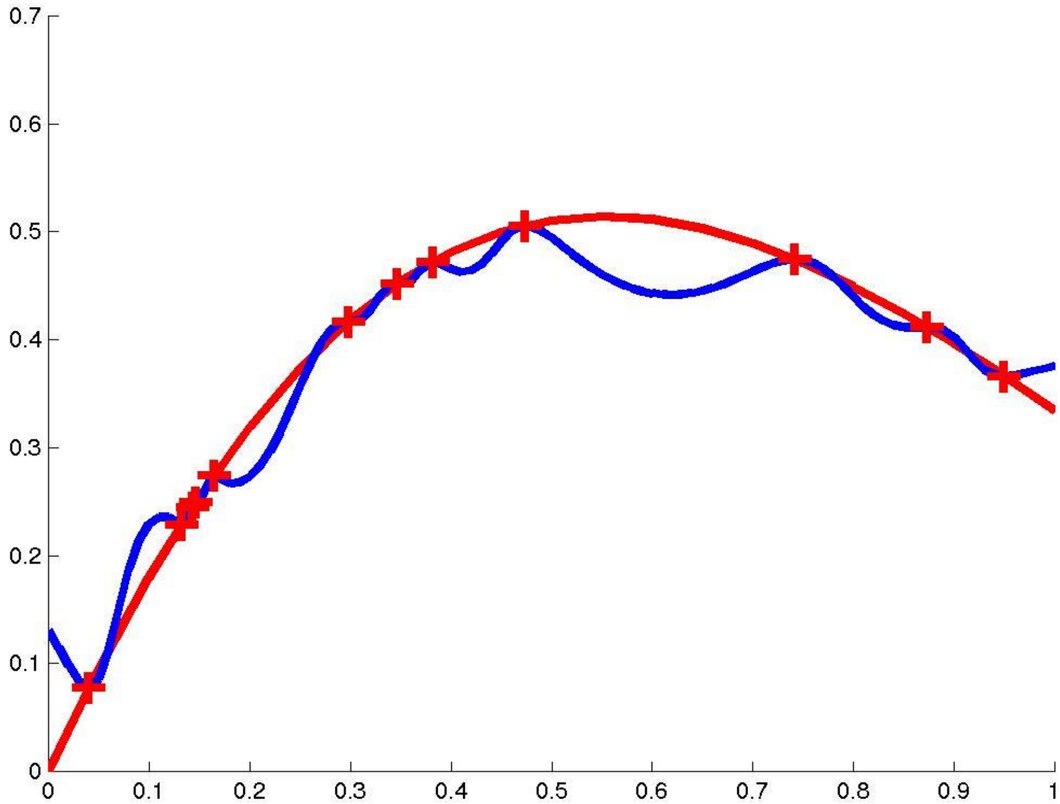

1-nearest neighbor
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

Recall


5-nearest neighbor
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

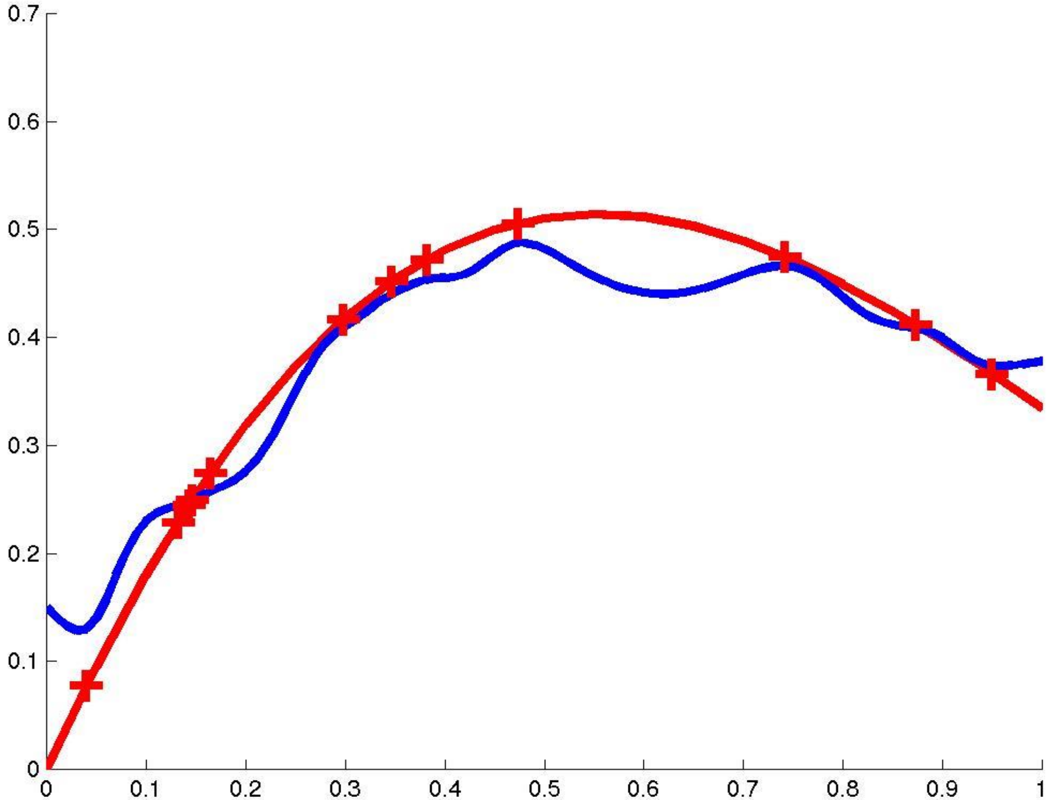
Distance Weighted NN

$W_i(\mathrm{d}(x_q, x_i)) = 1/\,\mathrm{d}(x_q, x_i)^2$



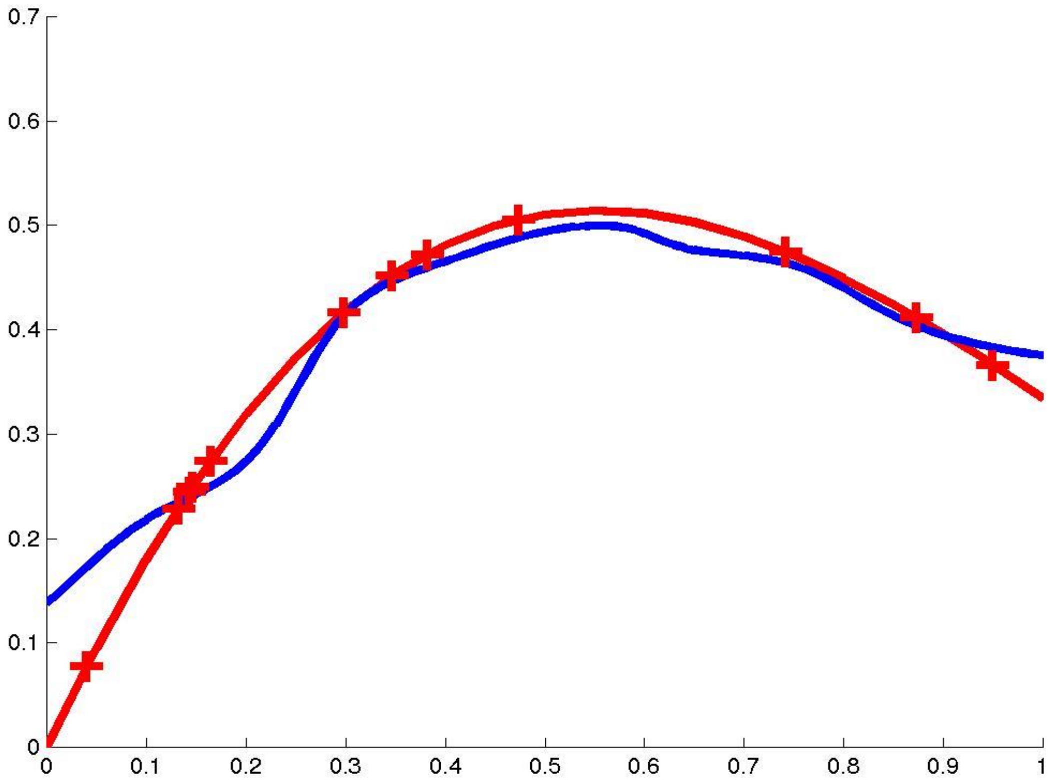
Distance Weighted NN

$$W_i(\mathrm{d}(x_q, x_i)) = 1/(d_0+\mathrm{d}(x_q,x_i))^2$$



Distance Weighted NN

$$W_i(d(x_q, x_i)) = e^{-(d(x_q, x_i)/\sigma_0)^2}$$



Overview:

A memory based learner – 4 factors

A memory based learner: 4 factors

1. A distance metric
2. How many nearby neighbors to look at?
3. A weighting function (optional)
4. How to fit with the local points?

1-NN

A memory based learner: 4 factors

1. A distance metric

   Euclidian

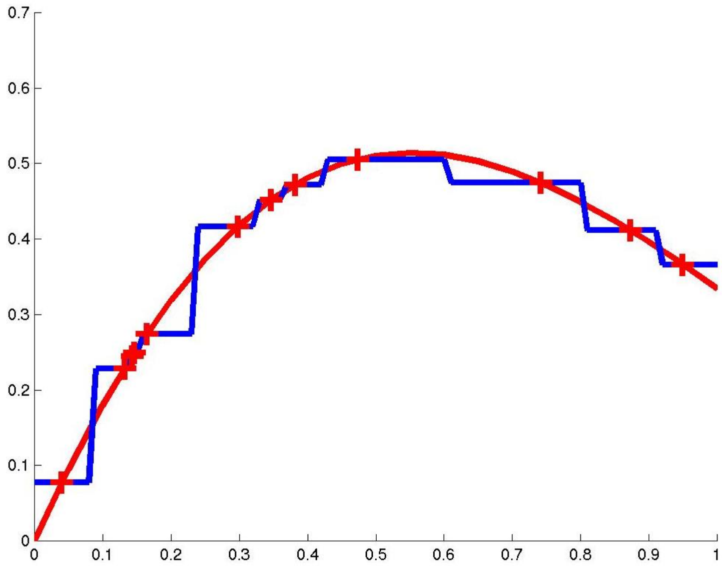
2. How many nearby neighbors to look at? One

3. A weighting function (optional)

   Unused

4. How to fit with the local points?

the same as the nearest neighbor.



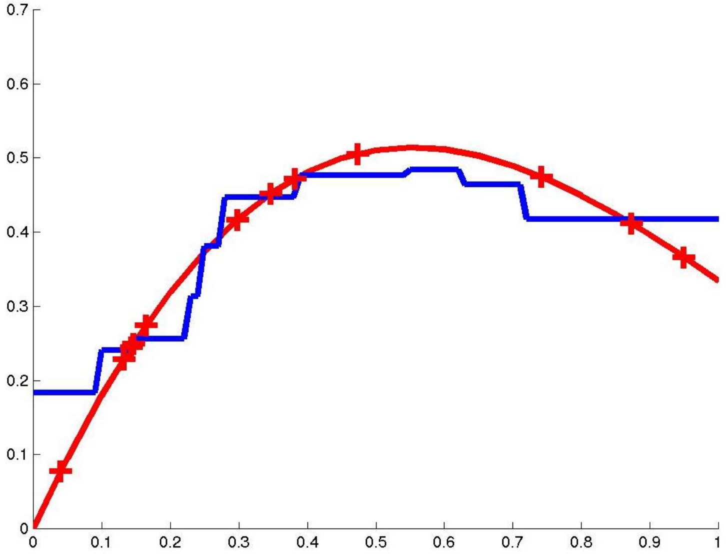
K-NN

A memory based learner: 4 factors

1. A distance metric

   Euclidian

2. How many nearby neighbors to look at? K



3. A weighting function (optional)

   Unused

4. How to fit with the local points?

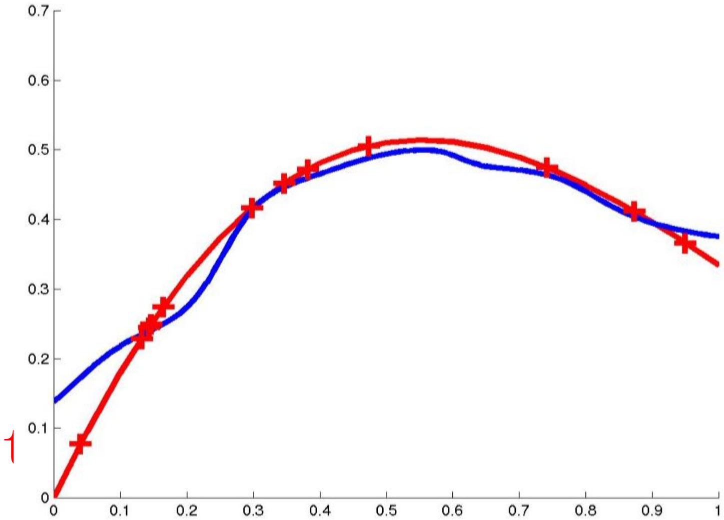
Voting among K neighbors

Distance-weighted KNN

A memory based learner: 4 factors

1. A distance metric
   Euclidian
2. How many nearby neighbors
   to look at? All of them, or K
3. A weighting function (optional)
   e. g. $w_i = exp(-D(x_i, query)^2 / K_w^2)$
   K_w: Kernel Width. Very important
4. How to fit with the local points?
   the weighted average of the outputs
   predict = $\Sigma w_i y_i / \Sigma w_i$

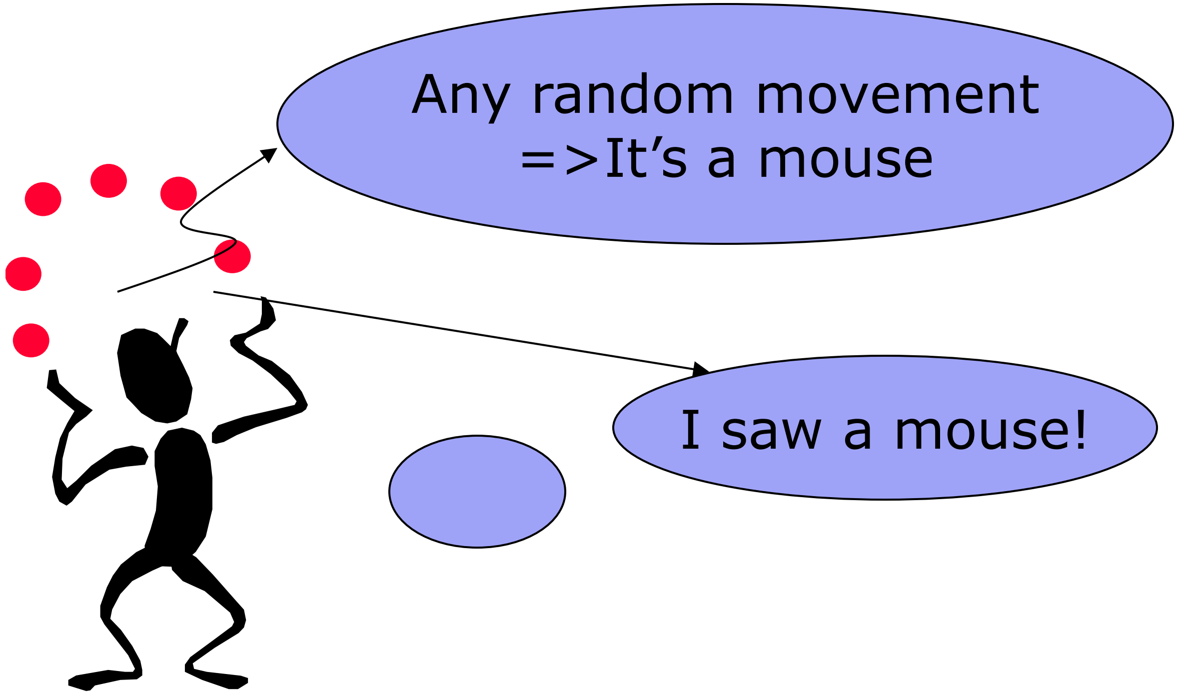
$$W_i(d(x_q, x_i)) = e^{-(d(x_q, x_i)/\sigma_0)^2}$$



Lazy and eager learning

Different learning methods

- Eager Learning



Different learning methods

- Instance-based Learning (lazy learning)

Lazy vs. Eager Learning

- Lazy: wait for query before generalizing
  - Training time: short
  - Test time: time consuming
- Lazy learner:
  - Can create local approximations

- Eager: generalize before seeing query
  - Training time: long
  - Test time: short
- Eager learner:
  - Use same model to each query
  - Tend to create global approximation

If they use the same hypothesis space, lazy can represent more complex functions