Introduction to Machine Learning
Similarity-based Learning Models

Prof. Chang-Chieh Cheng
Information Technology Service Center
National Chiao Tung University

Linear Algebra: Vectors

- Using a vector to represent a set of numbers

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

  - $\mathbf{x}$ is an n-dimension vector
  - In data analysis, we can use a vector to describe a data instance that consist of a set of features

Linear Algebra: Vectors

- The speed and agility ratings for 20 college athletes labelled with the decisions for whether they were drafted or not.
- Two features:
  - **Speed**
  - **Agility**

ID	Speed	Agility	Draft	ID	Speed	Agility	Draft
1	2.50	6.00	No	11	2.00	2.00	No
2	3.75	8.00	No	12	5.00	2.50	No
3	2.25	5.50	No	13	8.25	8.50	No
4	3.25	8.25	No	14	5.75	8.75	Yes
5	2.75	7.50	No	15	4.75	6.25	Yes
6	4.50	5.00	No	16	5.50	6.75	Yes
7	3.50	5.25	No	17	5.25	9.50	Yes
8	3.00	3.25	No	18	7.00	4.25	Yes
9	4.00	4.00	No	19	7.50	8.00	Yes
10	4.25	3.75	No	20	7.25	5.75	Yes

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Linear Algebra: Vectors

- $\mathbf{x_2}$ and $\mathbf{x_{10}}$ are two college athletics of ID = 2 and ID = 10

$$\mathbf{x_2} = \begin{bmatrix} 3.75 \\ 8.00 \end{bmatrix}$$

$$\mathbf{x_{10}} = \begin{bmatrix} 4.25 \\ 3.75 \end{bmatrix}$$

Linear Algebra: Vectors

- Weather observation

Day	MinTemp	MaxTemp	Humidity	Rain
1	24	31	0.7	0.4
2	24	33	0.65	0.3
3	23	30	0.6	0.3
4	24	32	0.8	0.7
5	23	29	0.8	0.8
6	22	29	0.7	0.6
7	21	27	0.6	0.2

Linear Algebra: Vectors

- $\mathbf{x_3}$ and $\mathbf{x_6}$ are data instances of day 3 and day 6

$$\mathbf{x_3} = \begin{bmatrix} 23 \\ 30 \\ 0.6 \\ 0.3 \end{bmatrix}$$

$$\mathbf{x_6} = \begin{bmatrix} 22 \\ 29 \\ 0.7 \\ 0.6 \end{bmatrix}$$

Linear Algebra: Distance

- Difference between two vectors
  - $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$
  - $\mathbf{x} - \mathbf{y} = \begin{bmatrix} x_1 - y_1 \\ x_2 - y_2 \\ \vdots \\ x_n - yn \end{bmatrix}$
- How to quantize the difference? How do we estimate a number to describe the difference?
  - For measurement
  - For Comparison

Linear Algebra: Distance

- Given two vectors of n-dimension, $\mathbf{x}$ and $\mathbf{y}$, a distance function d must conform to the following four criteria:
  - Non-negativity

    $d(\mathbf{x}, \mathbf{y}) \geq 0$
  - Identity

    $d(\mathbf{x}, \mathbf{y}) = 0 \leftrightarrow \mathbf{x} = \mathbf{y}$
  - Symmetry

    $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$
  - Triangular Inequality

    $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{y}, \mathbf{z})$

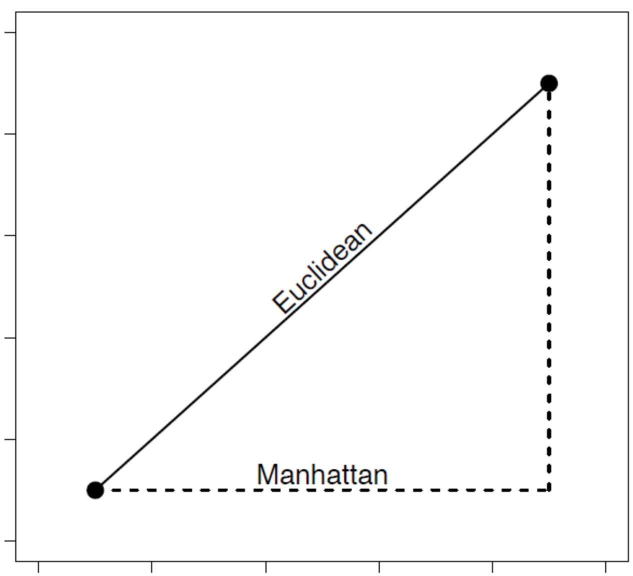
- We can use a distance function to describe **similarity** of two vectors

Linear Algebra: Distance

- L^2-norm
- $$\|\mathbf{x}\|_2 = \sqrt{|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2} = \sqrt{\sum_{i=1}^{n} |x_i|^2}$$
  - Euclidean distance
  - Euclidean norm
  - $\|\mathbf{x}\|_2 = \sqrt{\mathbf{x} \cdot \mathbf{x}}$
  - $\frac{\partial \|\mathbf{x}\|_2}{\partial \mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|_2}$
  - $\frac{\partial \|\mathbf{x}\|_2}{\partial x_i} = \frac{x_i}{\|\mathbf{x}\|_2}, 1 \le i \le n$

Linear Algebra: Distance

- L^1-norm
- $$\|\mathbf{x}\|_1 = |x_1| + |x_2| + \cdots + |x_n| = \sum_{i=1}^{n} |x_i|$$
  - Manhattan distance



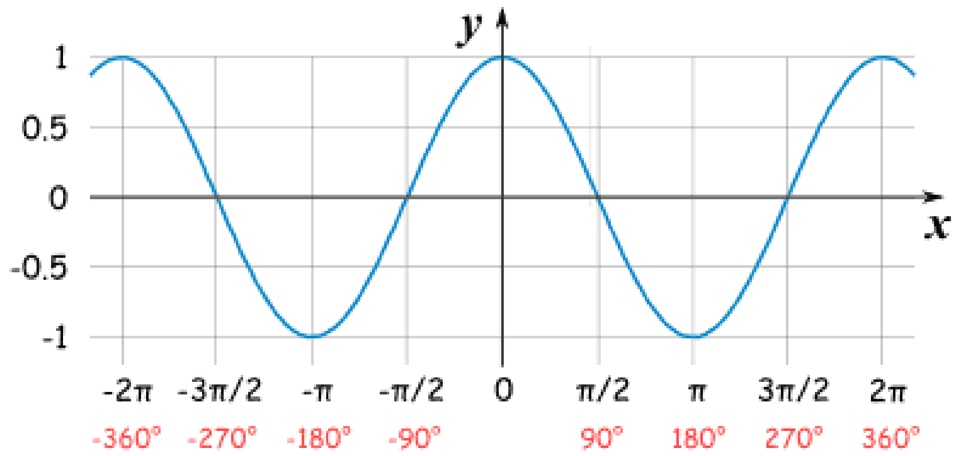
Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Linear Algebra: Distance

- $\cos\theta$

$$\mathbf{x}^{\mathbf{T}}\mathbf{y} = \quad \|\mathbf{x}\|_2 \|\mathbf{y}\|_2 \cos \boldsymbol{\theta}$$

  - where θ is the angle between **x** and **y**.



Linear Algebra: Range Normalization

- A dataset listing the salary and age information for customers and whether or not the purchased a pension plan

ID	Salary	Age	Purchased
1	53700	41	No
2	65300	37	No
3	48900	45	Yes
4	64800	49	Yes
5	44200	30	No
6	55900	57	Yes
7	48600	26	No
8	72800	60	Yes
9	45300	34	No
10	73200	52	Yes

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Linear Algebra: Range Normalization

- $\mathbf{x_1}$, $\mathbf{x_8}$, and $\mathbf{x_{10}}$ are three data instances of ID = 1, ID = 6, and ID 10.

$$\mathbf{x_1} = \begin{bmatrix} 53700 \\ 41 \end{bmatrix}, \mathbf{x_8} = \begin{bmatrix} 72800 \\ 60 \end{bmatrix}, \mathbf{x_{10}} = \begin{bmatrix} 73200 \\ 52 \end{bmatrix}$$

$$\mathbf{x_1} - \mathbf{x_8} = \begin{bmatrix} -19100 \\ -19 \end{bmatrix}$$

$$\|\mathbf{x_1} - \mathbf{x_8}\|_2 \approx 19100.01$$

$$\|\mathbf{x_1} - \mathbf{x_8}\|_1 = 19119$$

$\cos\theta = 0.099$

$\mathbf{x_1}$ is more similar to $\mathbf{x_8}$ than $\mathbf{x_{10}}$

$$\mathbf{x_1} - \mathbf{x_{10}} = \begin{bmatrix} -19500 \\ -11 \end{bmatrix}$$

$$\|\mathbf{x_1} - \mathbf{x_{10}}\|_2 \approx 19500.003$$

$$\|\mathbf{x_1} - \mathbf{x_{10}}\|_1 = 19511$$

$\cos\theta = 0.093$

Linear Algebra: Range Normalization

- Range normalization
  - Given a value x in a range of $[x_{min}, x_{max}]$, the normalized value of x, x', can be estimated by the following equation

$$\boldsymbol{x' = \frac{x - x_{min}}{x_{max} - x_{min}}(x'_{max} - x'_{min}) + x'_{min}}$$

    - where $[x'_{min}, x'_{max}]$ is the range of normalization. In the most cases, $[x'_{min}, x'_{max}]$ = [0.0, 1.0], [-0.5, 0.5], or [-1.0, 1.0]

Linear Algebra: Range Normalization

- Normalized data

Normalized Dataset			
ID	Salary	Age	Purch.
1	0.3276	0.4412	No
2	0.7276	0.3235	No
3	0.1621	0.5588	Yes
4	0.7103	0.6765	Yes
5	0.0000	0.1176	No
6	0.4034	0.9118	Yes
7	0.1517	0.0000	No
8	0.9862	1.0000	Yes
9	0.0379	0.2353	No
10	1.0000	0.7647	Yes

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Linear Algebra: Range Normalization

-

$$\mathbf{x_1} = \begin{bmatrix} 0.3276 \\ 0.4412 \end{bmatrix}, \mathbf{x_8} = \begin{bmatrix} 0.9862 \\ 1.0 \end{bmatrix}, \mathbf{x_{10}} = \begin{bmatrix} 1.0 \\ 0.7647 \end{bmatrix}$$

$$\mathbf{x_1} - \mathbf{x_8} = \begin{bmatrix} -0.6586 \\ -0.5588 \end{bmatrix}$$

$$\|\mathbf{x_1} - \mathbf{x_8}\|_2 \approx 0.864$$

$$\|\mathbf{x_1} - \mathbf{x_8}\|_1 \approx 1.21$$

$\cos\theta = 0.99$

$$\mathbf{x_1} - \mathbf{x_{10}} = \begin{bmatrix} -0.6724 \\ -0.3235 \end{bmatrix}$$

$$\|\mathbf{x_1} - \mathbf{x_{10}}\|_2 \approx 0.746$$

$$\|\mathbf{x_1} - \mathbf{x_{10}}\|_1 \approx 0.9959$$

$\cos\theta = 0.96$

L1 and L2:

$\mathbf{x_1}$ is more similar to $\mathbf{x_{10}}$ than $\mathbf{x_8}$

cos*θ*:

$\mathbf{x_1}$ is more similar to $\mathbf{x_8}$ than $\mathbf{x_{10}}$

The Nearest Neighbor

- Input:
  - Training dataset
  - A **query** to be classified
- Algorithm
  1. Iterate across the instances in memory and find the instance that is shortest distance from the query position in the feature space.
  2. Make a prediction for the query equal to the value of the target feature of the nearest neighbor

The Nearest Neighbor

- Training dataset:

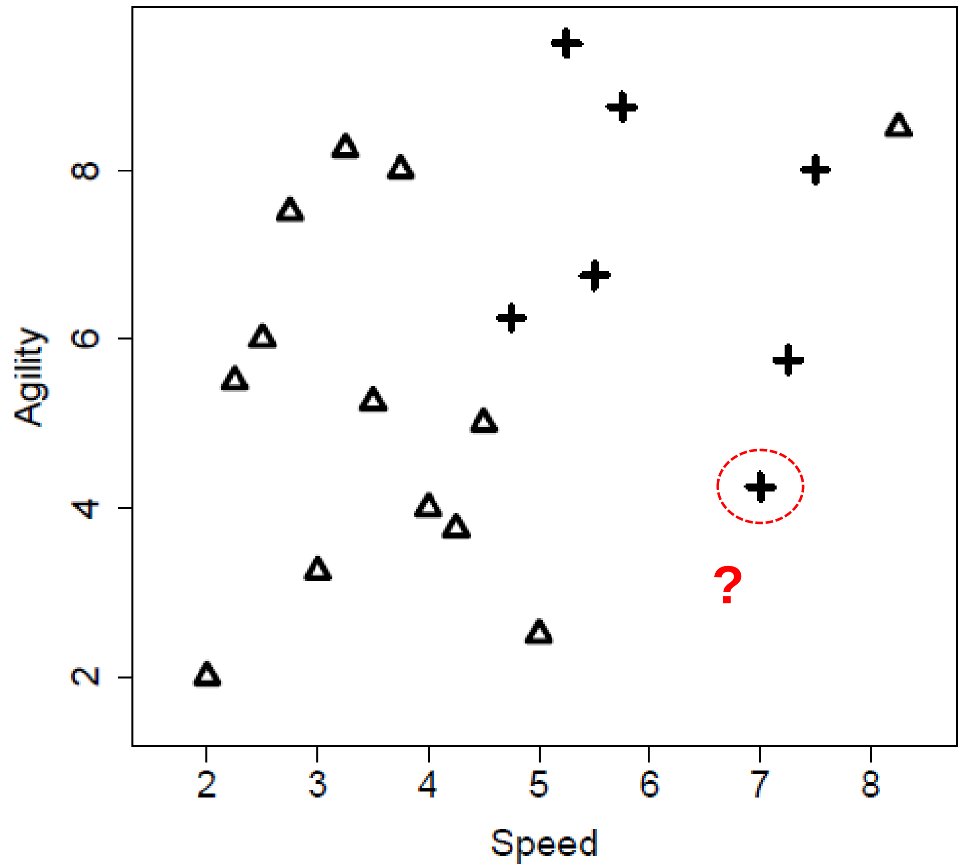
ID	Speed	Agility	Draft	ID	Speed	Agility	Draft
1	2.50	6.00	No	11	2.00	2.00	No
2	3.75	8.00	No	12	5.00	2.50	No
3	2.25	5.50	No	13	8.25	8.50	No
4	3.25	8.25	No	14	5.75	8.75	Yes
5	2.75	7.50	No	15	4.75	6.25	Yes
6	4.50	5.00	No	16	5.50	6.75	Yes
7	3.50	5.25	No	17	5.25	9.50	Yes
8	3.00	3.25	No	18	7.00	4.25	Yes
9	4.00	4.00	No	19	7.50	8.00	Yes
10	4.25	3.75	No	20	7.25	5.75	Yes

- Query:
  - Should we draft this guy?
    - **SPEED= 6.75, AGILITY= 3**

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

The Nearest Neighbor

- Figure of the training dataset
  - Δ: Draft = NO
  - +: Draft = YES
  - ?: the query



Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

The Nearest Neighbor

- The L2 distances between the training data and the query

ID	SPEED	AGILITY	DRAFT	Dist.
18	7.00	4.25	yes	1.27
12	5.00	2.50	no	1.82
10	4.25	3.75	no	2.61
20	7.25	5.75	yes	2.80
9	4.00	4.00	no	2.93
6	4.50	5.00	no	3.01
8	3.00	3.25	no	3.76
15	4.75	6.25	yes	3.82
7	3.50	5.25	no	3.95
16	5.50	6.75	yes	3.95

ID	SPEED	AGILITY	DRAFT	Dist.
11	2.00	2.00	no	4.85
19	7.50	8.00	yes	5.06
3	2.25	5.50	no	5.15
1	2.50	6.00	no	5.20
13	8.25	8.50	no	5.70
2	3.75	8.00	no	5.83
14	5.75	8.75	yes	5.84
5	2.75	7.50	no	6.02
4	3.25	8.25	no	6.31
17	5.25	9.50	yes	6.67

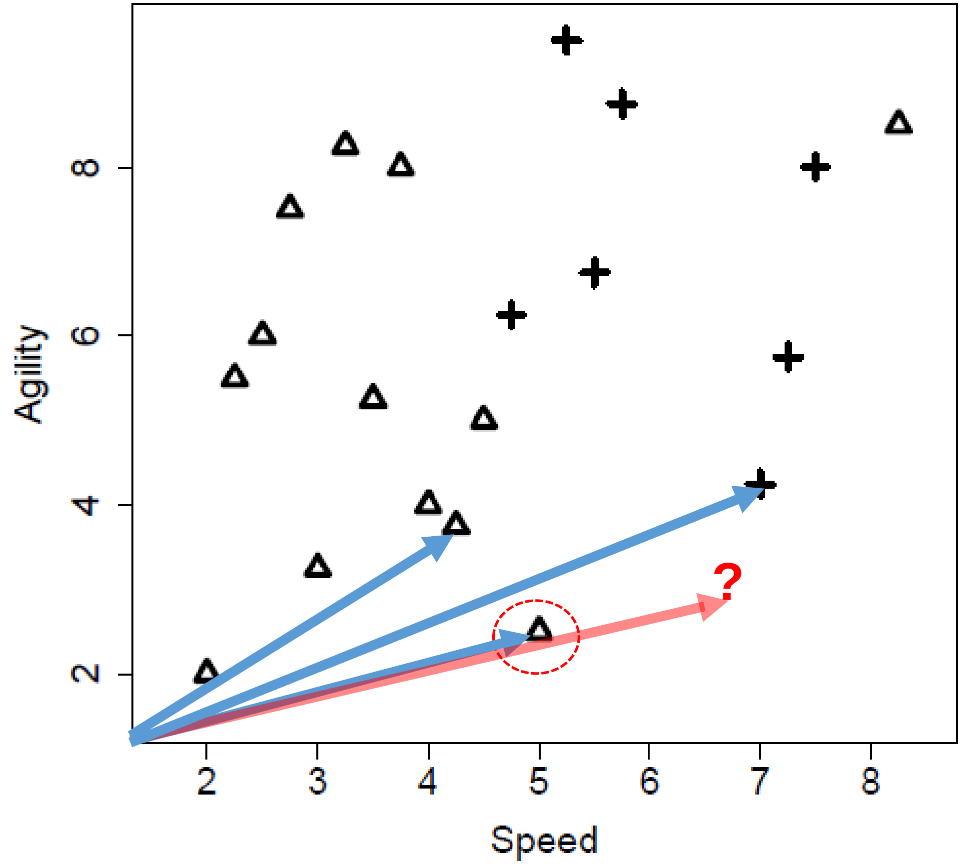
Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

The Nearest Neighbor

- The L1 distances
  - **ID 18: 1.5**
  - ID 12: 2.25
  - ID 10: 3.25

The Nearest Neighbor

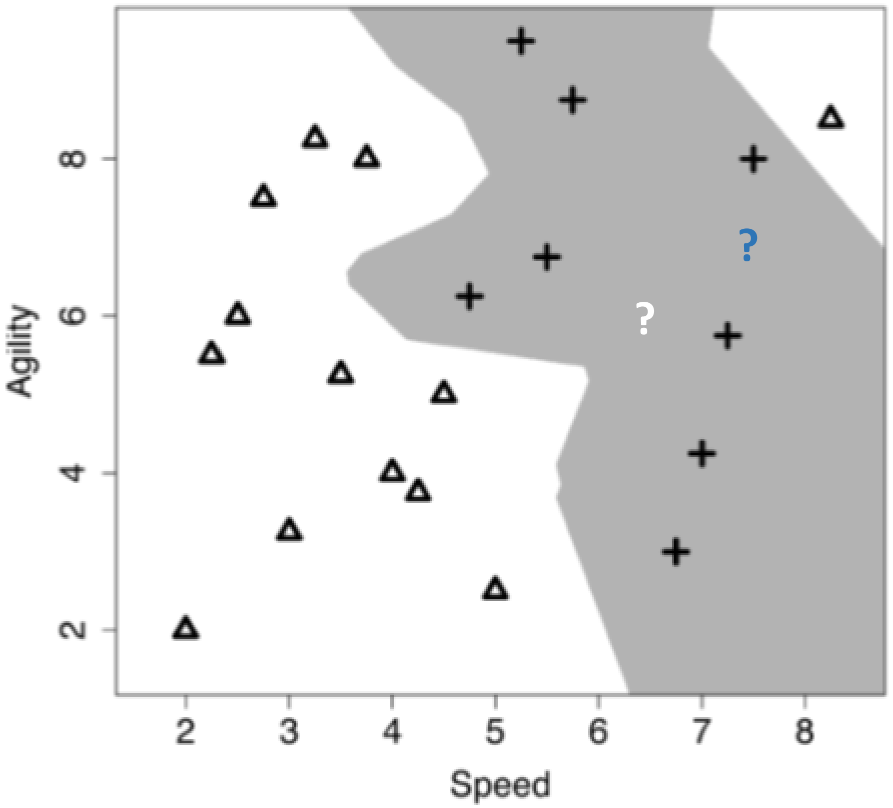
- The cosine distances
  - **ID 12: 0.999**
  - ID 18: 0.992
  - ID 10: 0.954



Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

The Nearest Neighbor

- Noisy
  - Query 1(blue ?):
    - Speed = 8.0
    - Agility = 8.0
    - ➔The NN is ID13 (8.25, 8.50)
    - ➔Draft = No
  - Query 2 (white ?):
    - Speed = 7.0
    - Agility = 7.0
    - ➔The NN is ID19 (7.5, 8.0)
    - ➔Draft = Yes



Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

The K Nearest Neighbors, KNN

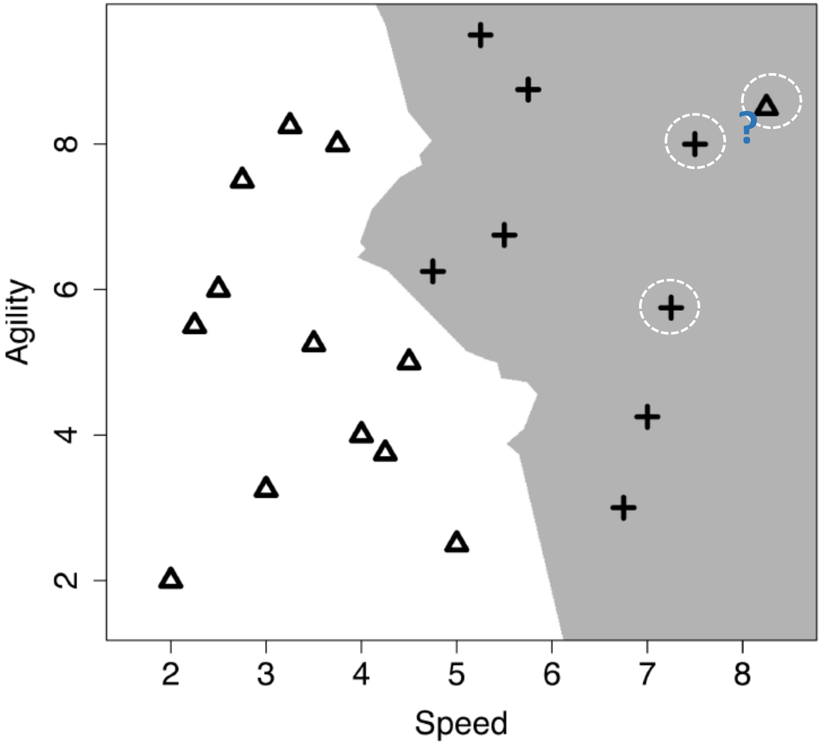
- KNN predicts the target level with the majority vote from the set of *k* nearest neighbors to the query **q**:

$$\mathbf{M}_{\boldsymbol{k}}(\mathbf{q}) = \mathbf{arg} \max_{\boldsymbol{l} \in \boldsymbol{levels}(\boldsymbol{t})} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{k}} \boldsymbol{\delta}(\boldsymbol{t}_{\boldsymbol{i}}, \boldsymbol{l})$$

  - where $\boldsymbol{\delta}(\boldsymbol{t}_{\boldsymbol{i}}, \boldsymbol{l}) = \mathbf{1}$ if t_i equals to l and 0 otherwise

The K Nearest Neighbors, KNN

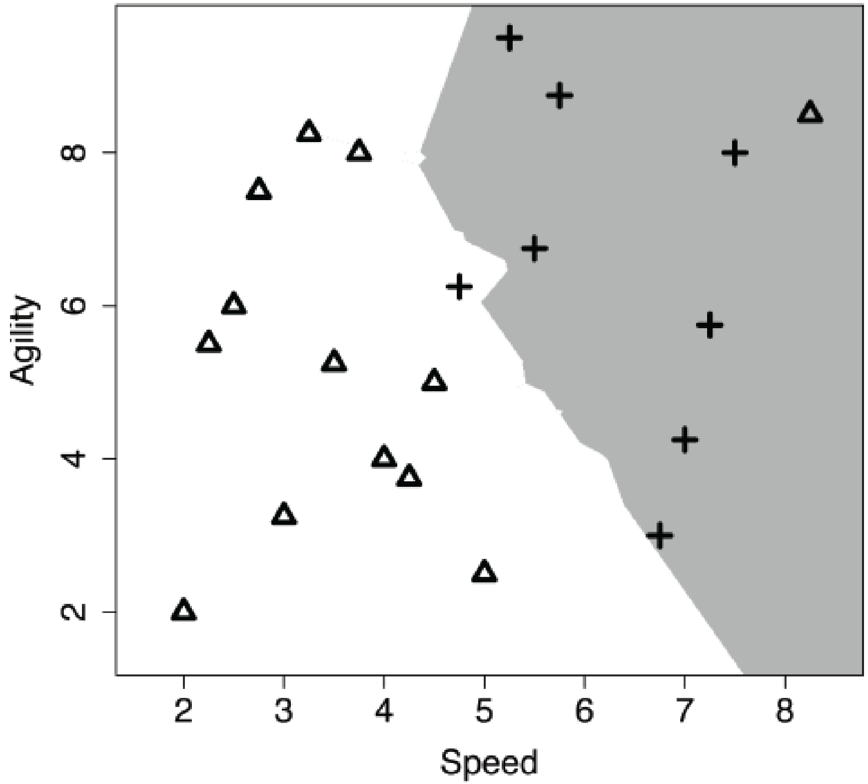
- *K* = 3
  - Query: Speed = 8.0, Agility = 8.0
    - Neighbors:
      - ID13 (8.25, 8.50, NO)
      - ID19 (7.5, 8.0, Yes)
      - ID13 (7.25, 5.75, Yes)
    - 2 Yes: 1 No
    - ➔Draft = Yes



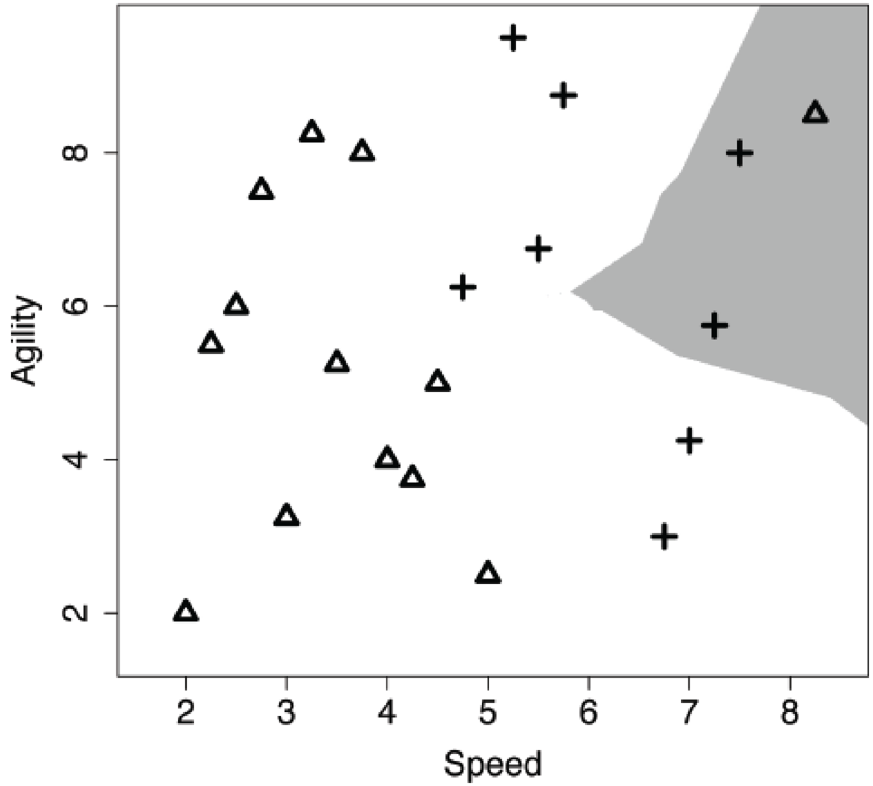
Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

The K Nearest Neighbors, KNN

- K = 5



K = 15



Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

The K Nearest Neighbors, KNN

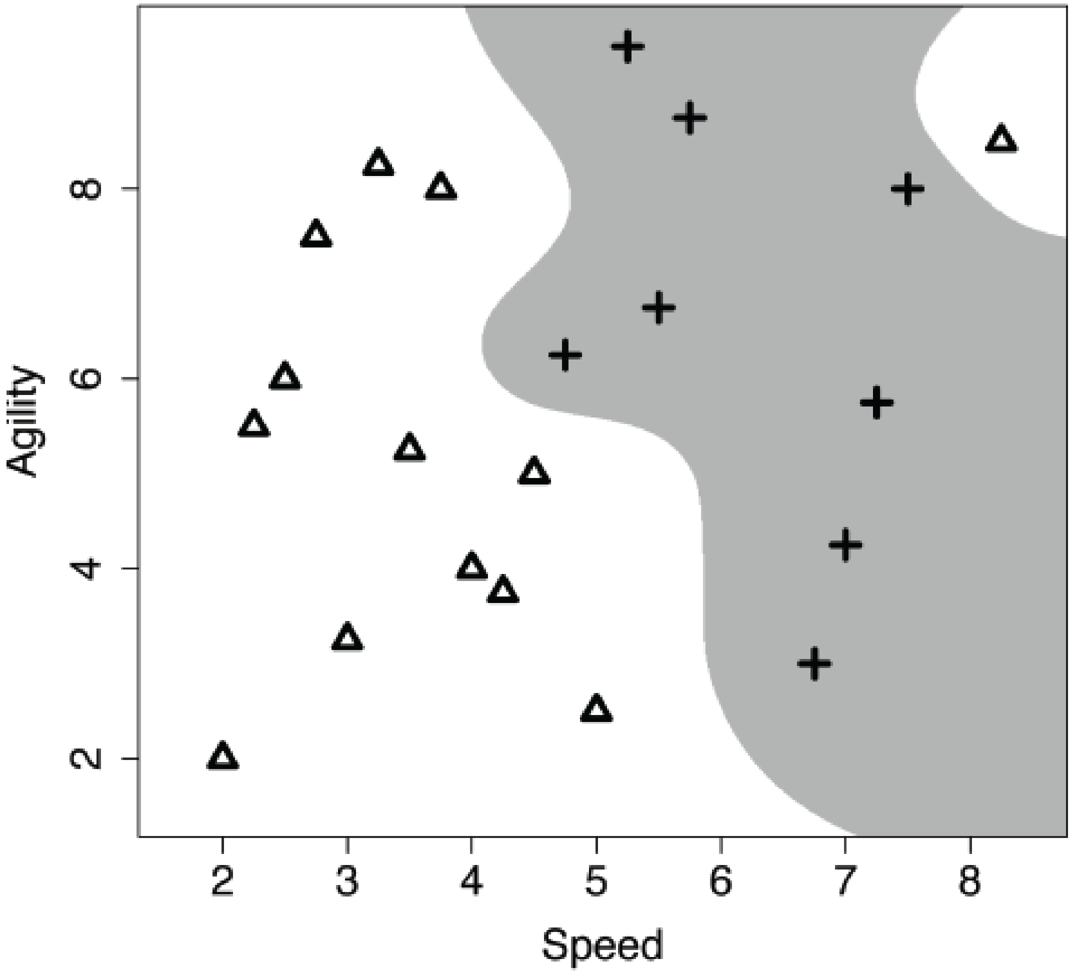
- Weighted KNN

$$\mathbf{M}_k(\mathbf{q}) = \mathbf{arg} \max_{l \in levels(t)} \sum_{i=1}^{k} \frac{1}{d(\mathbf{q}, \mathbf{x}_i)^2} \delta(t_i, l)$$

  - where $d(\mathbf{q}, \mathbf{x}_i)$ is a distance function

The K Nearest Neighbors, KNN

- Weighted KNN
  - K = 21



Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

The K Nearest Neighbors, KNN

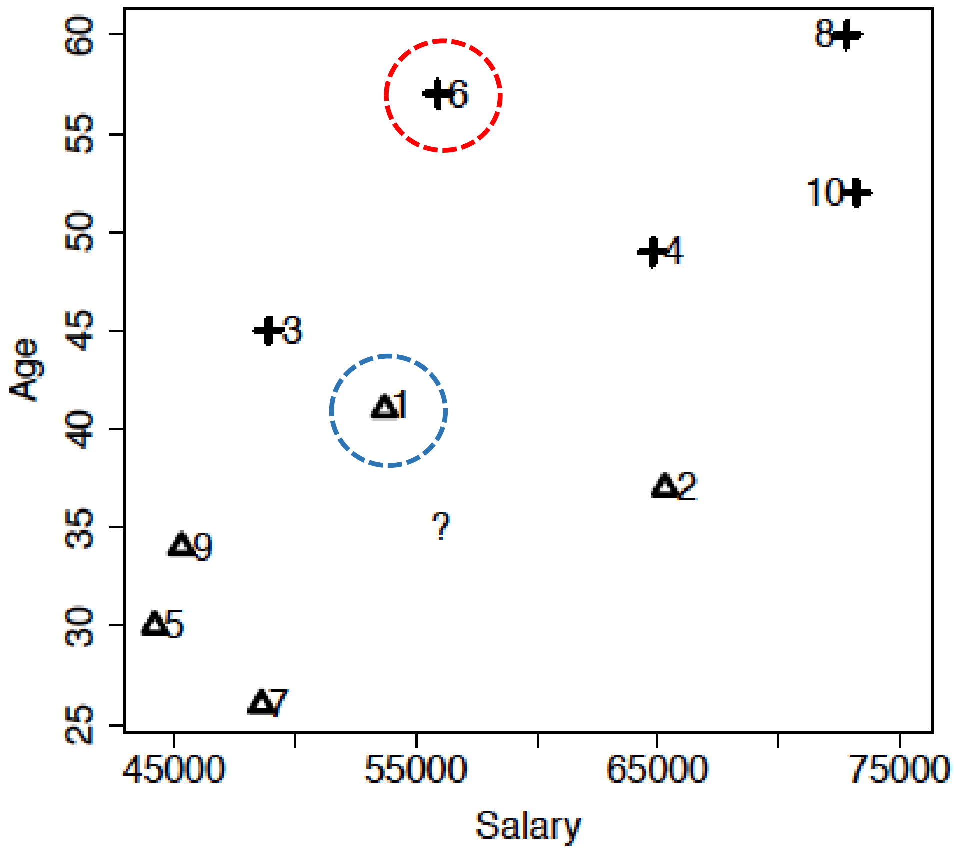
- KNN with data normalization
  - Customer dataset:

ID	Salary	Age	Purchased
1	53700	41	No
2	65300	37	No
3	48900	45	Yes
4	64800	49	Yes
5	44200	30	No
6	55900	57	Yes
7	48600	26	No
8	72800	60	Yes
9	45300	34	No
10	73200	52	Yes

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

The K Nearest Neighbors, KNN

- KNN without data normalization
  - Δ: Purchase = NO
  - +: Purchase = YES
  - ?: the query **q** = (56000, 35)
  - L^2-norm distance:
    - $d(\mathbf{q}, \mathbf{x}_1) = 2300.0078$
    - $d(\mathbf{q}, \mathbf{x}_6) = 102.3914$



Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

The K Nearest Neighbors, KNN

- KNN without data normalization

ID	Salary	Age	Purch.	Salary and Age Dist.	Salary and Age Neigh.	Salary Only Dist.	Salary Only Neigh.	Age Only Dist.	Age Only Neigh.
1	53700	41	No	2300.0078	2	2300	2	6	4
2	65300	37	No	9300.0002	6	9300	6	2	2
3	48900	45	Yes	7100.0070	3	7100	3	10	6
4	64800	49	Yes	8800.0111	5	8800	5	14	7
5	44200	30	No	11800.0011	8	11800	8	5	5
6	55900	57	Yes	102.3914	1	100	1	22	9
7	48600	26	No	7400.0055	4	7400	4	9	3
8	72800	60	Yes	16800.0186	9	16800	9	25	10
9	45300	34	No	10700.0000	7	10700	7	1	1
10	73200	52	Yes	17200.0084	10	17200	10	17	8

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

The K Nearest Neighbors, KNN

- KNN with data normalization

	Normalized Dataset			Salary and Age		Salary Only		Age Only	
ID	Salary	Age	Purch.	Dist.	Neigh.	Dist.	Neigh.	Dist.	Neigh.
1	0.3276	0.4412	No	0.1935	1	0.0793	2	0.17647	4
2	0.7276	0.3235	No	0.3260	2	0.3207	6	0.05882	2
3	0.1621	0.5588	Yes	0.3827	5	0.2448	3	0.29412	6
4	0.7103	0.6765	Yes	0.5115	7	0.3034	5	0.41176	7
5	0.0000	0.1176	No	0.4327	6	0.4069	8	0.14706	3
6	0.4034	0.9118	Yes	0.6471	8	0.0034	1	0.64706	9
7	0.1517	0.0000	No	0.3677	3	0.2552	4	0.26471	5
8	0.9862	1.0000	Yes	0.9361	10	0.5793	9	0.73529	10
9	0.0379	0.2353	No	0.3701	4	0.3690	7	0.02941	1
10	1.0000	0.7647	Yes	0.7757	9	0.5931	10	0.50000	8

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

The K Nearest Neighbors, KNN

- In the most cases, the data should be normalized before the machine learning training
- Notice that the normalized data could lose the real data meaning

K-d tree

- k-d tree (k-dimensional tree)
  - A space-partitioning data structure for organizing points in a k-dimensional space.
  - Searching with multidimensional search key
  - A special case of binary space partitioning trees.

Bentley, J. L. "Multidimensional binary search trees used for associative searching". Communications of the ACM. 18 (9): 509, 1975

K-d tree

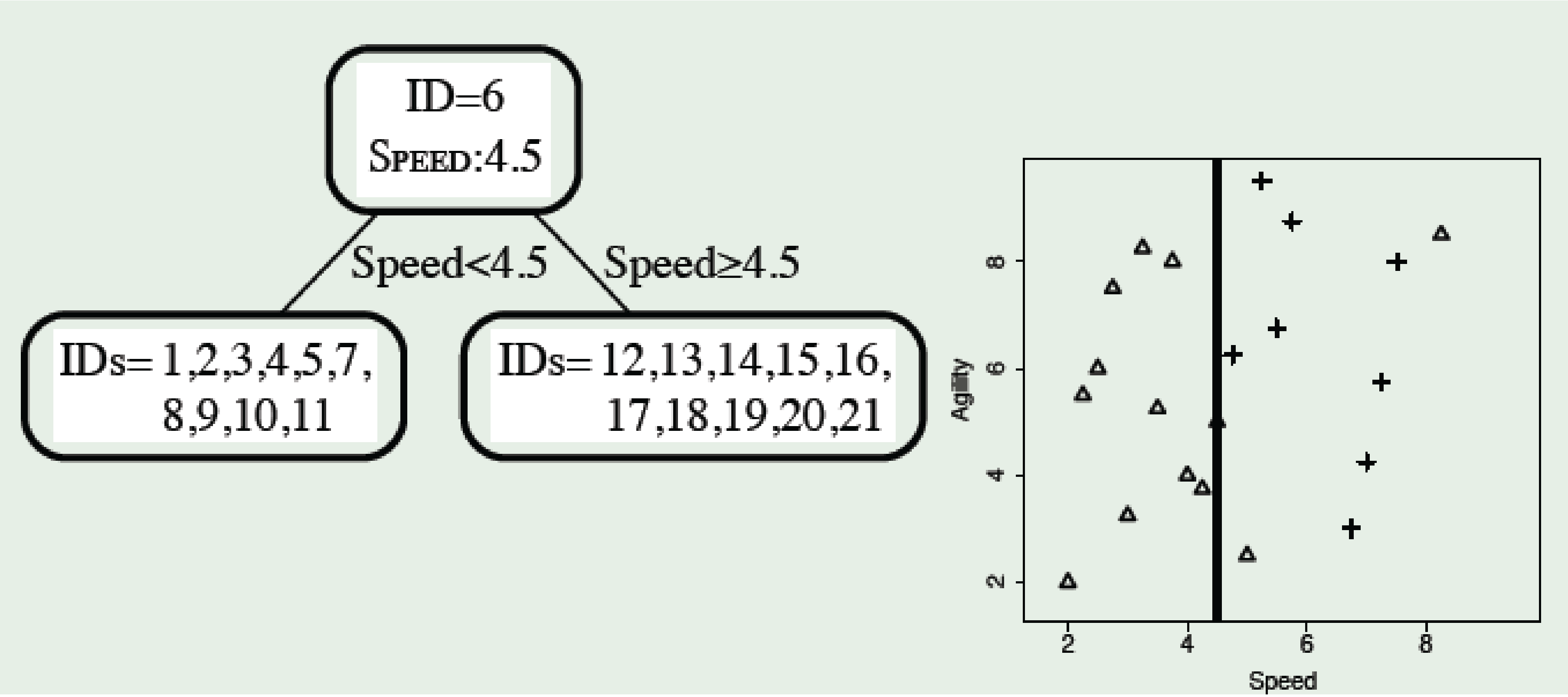
- Building a k-d tree
  - Dataset: college athletes
  - Dimension order:
    1. Speed
    2. Agility

ID	Speed	Agility	Draft
1	2.50	6.00	No
2	3.75	8.00	No
3	2.25	5.50	No
4	3.25	8.25	No
5	2.75	7.50	No
6	4.50	5.00	No
7	3.50	5.25	No
8	3.00	3.25	No
9	4.00	4.00	No
10	4.25	3.75	No
11	2.00	2.00	No
12	5.00	2.50	No
13	8.25	8.50	No
14	5.75	8.75	Yes
15	4.75	6.25	Yes
16	5.50	6.75	Yes
17	5.25	9.50	Yes
18	7.00	4.25	Yes
19	7.50	8.00	Yes
20	7.25	5.75	Yes
21	6.75	3.00	Yes

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

K-d tree

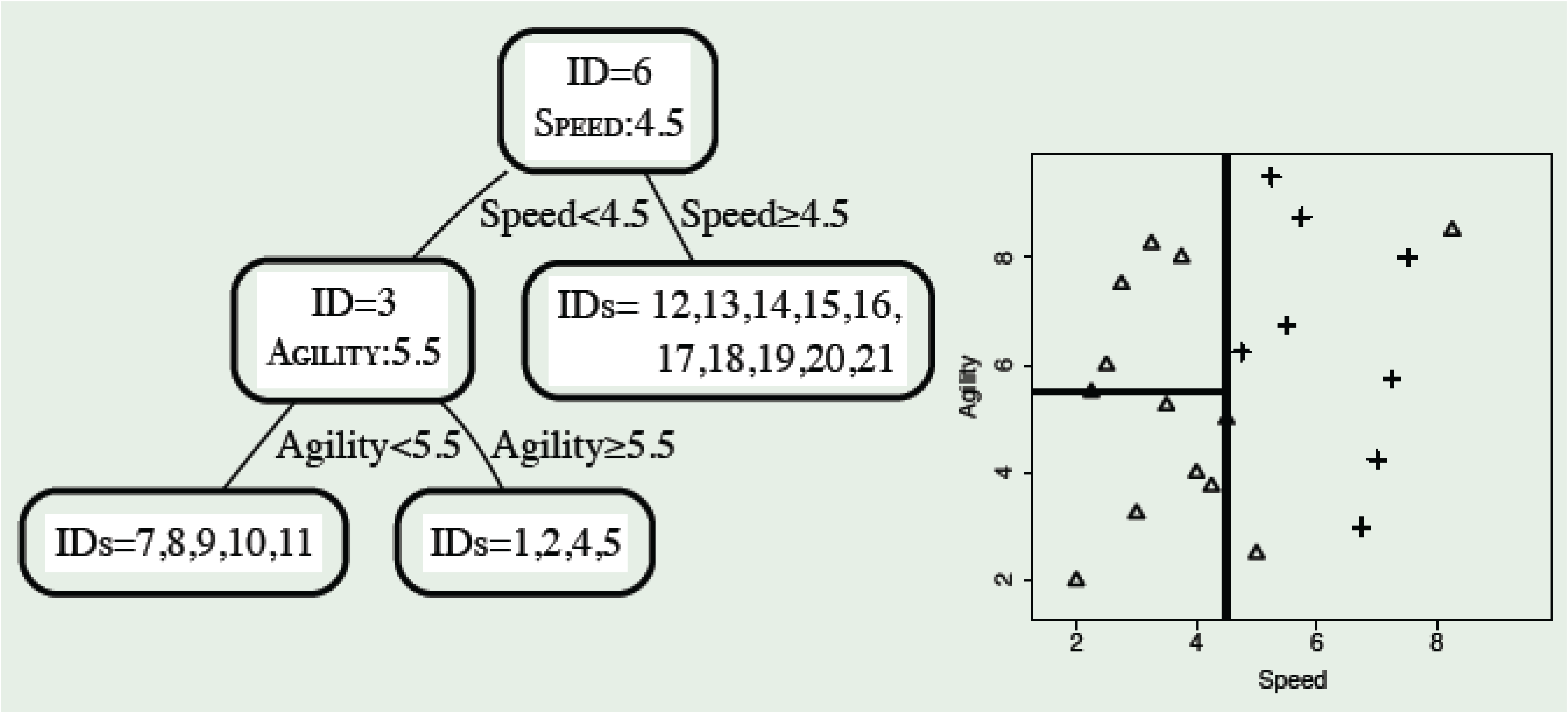
- Building a k-d tree
  - Splitting the first dimension: Speed
    - Using the median as the splitting pivot



Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

K-d tree

- Building a k-d tree
  - Splitting the second dimension: Agility



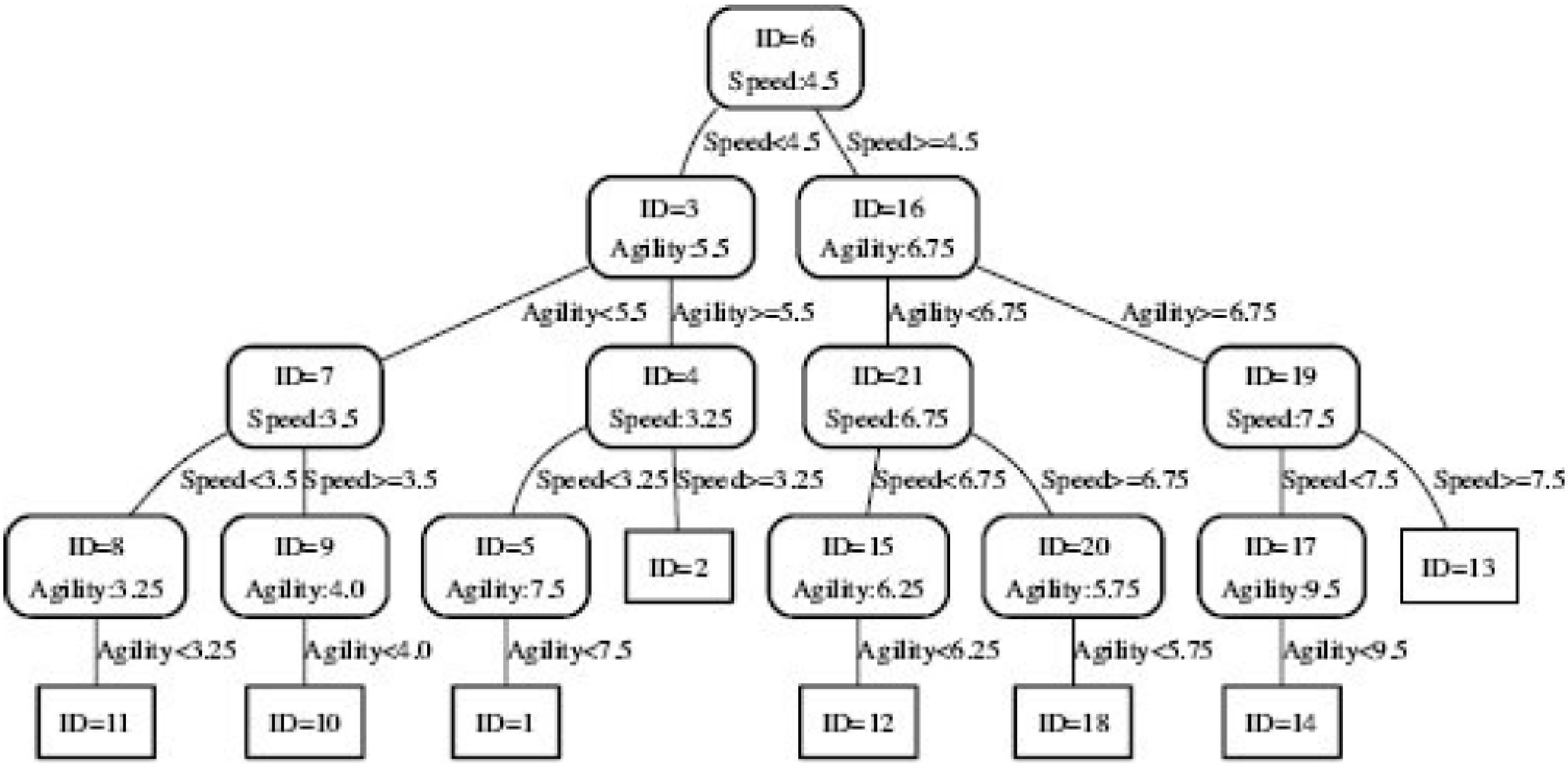
Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

K-d tree

- Building a k-d tree
    1. Splitting d_1
    2. Splitting d_2
    3. …
    4. Splitting d_k
    5. If the number of items in a cell > 1, Repeat step 1; otherwise stop.

K-d tree

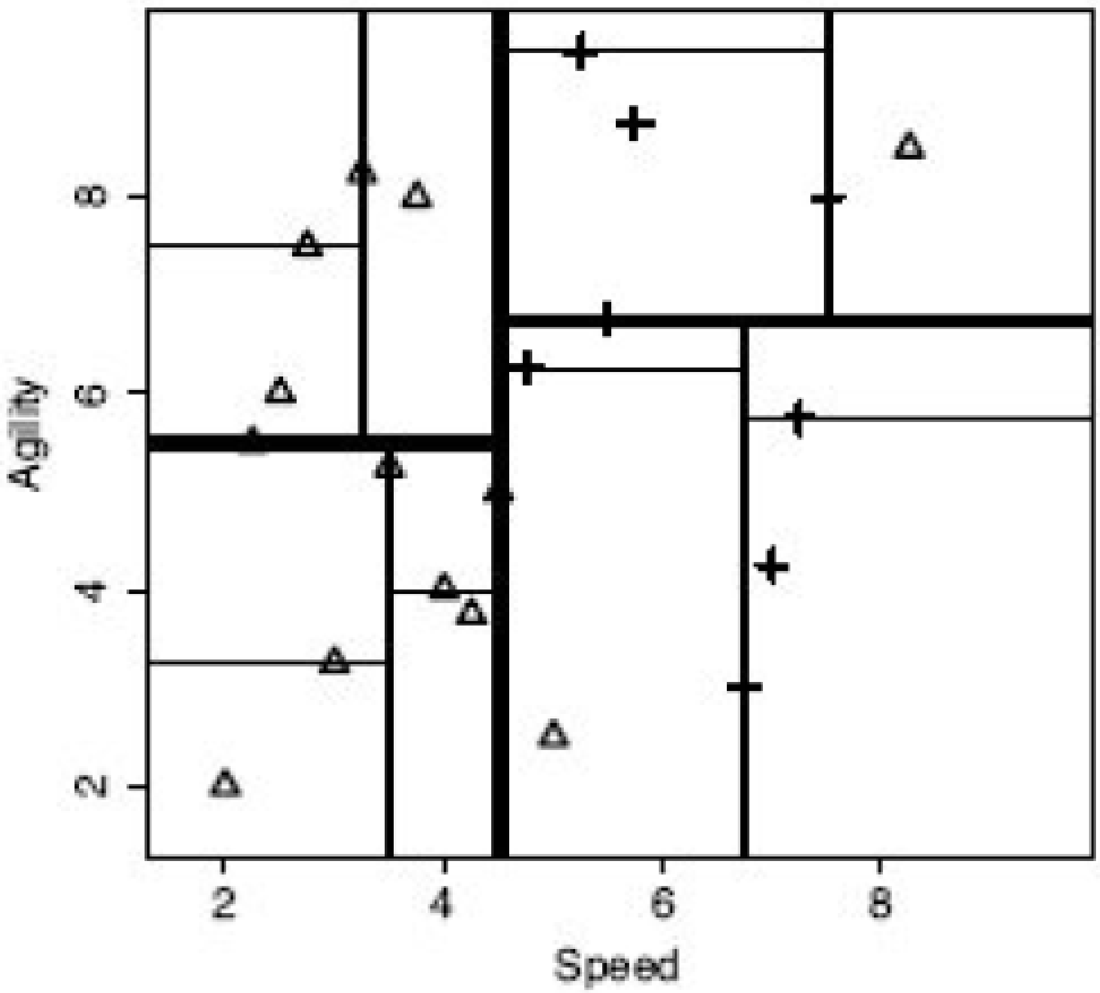
- Building a K-d tree
  - Final result



Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

K-d tree

- Building a K-d tree
  - Final result



Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

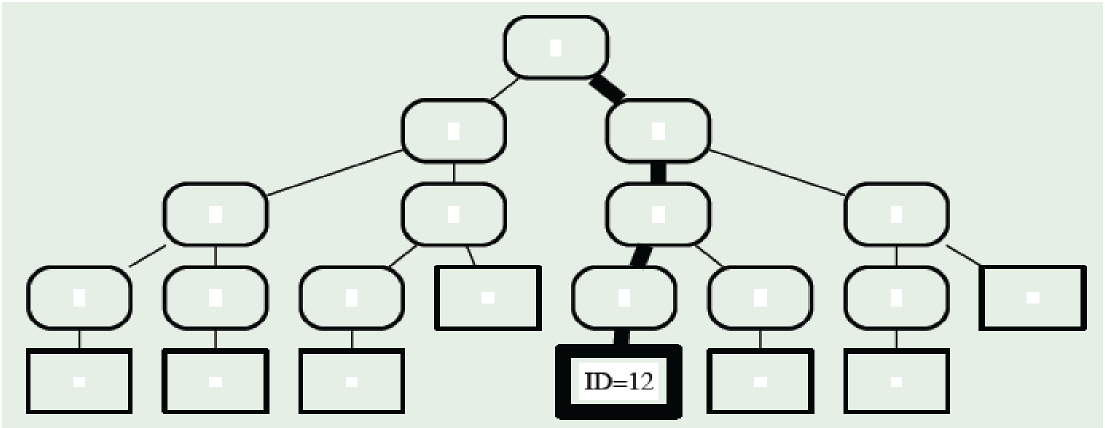
Finding the nearest neighbor by K-d tree

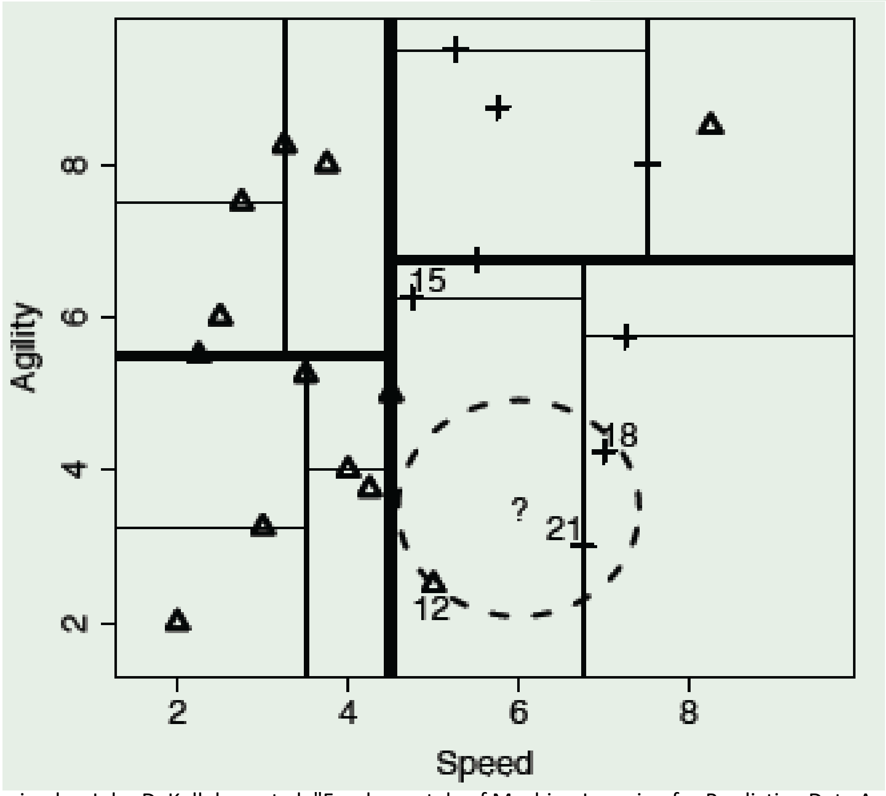
- Input:
  - Query: **q**
  - The root node of a K-d tree: **r**
- Algorithm
  1. The nearest neighbor, **t** = null
  2. The distance between **q** and **t,** $d_{\mathbf{t}} = \infty$
  3. **x** = **descendTree**(**r**, **q**)
  4. while **x** != NULL do
  5. if d(**q**, **x**) < $d_{\mathbf{t}}$ ➔ **t** = **x**; $d_{\mathbf{t}}$ = d(**q**, **x**)
  6. if **boundaryDist**(**q**, **x**) < $d_{\mathbf{t}}$ ➔ **x** = **descendTree**(**x**, **q**)
  7. else ➔ **x** = **parent**(**x**)
  8. return **t**

descendTree(**x**, q): simple search from the node **x** to leaf
parent(**x**): returns the parent node of **x** and prune **x**
boundaryDist(**q**, **x**): comparing the distance between **q** and the hyperplane of **x** if **x** is a non-leaf node; otherwise, returns false.

Finding the nearest neighbor by K-d tree

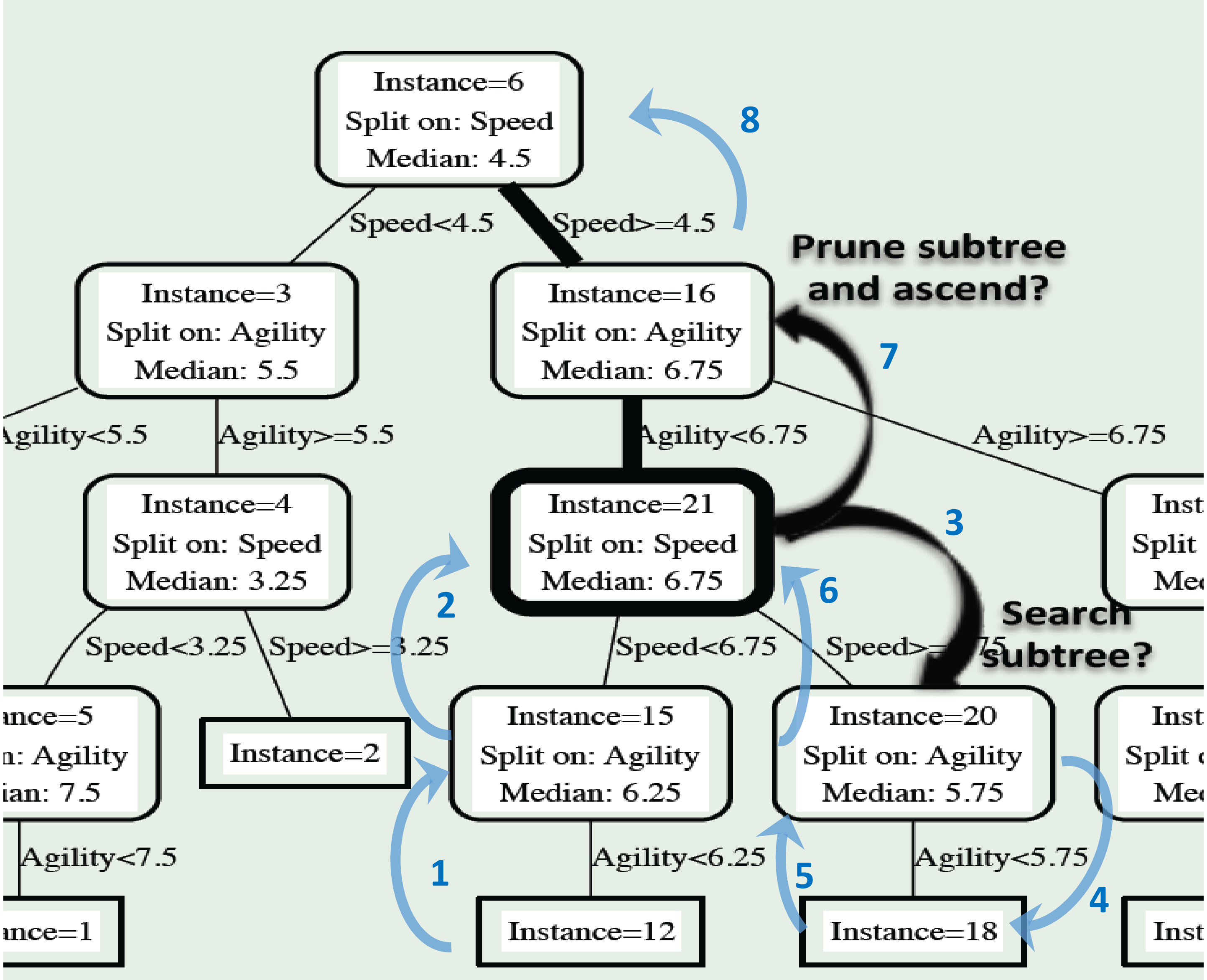
- **q** = (6.0, 3.5)





Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

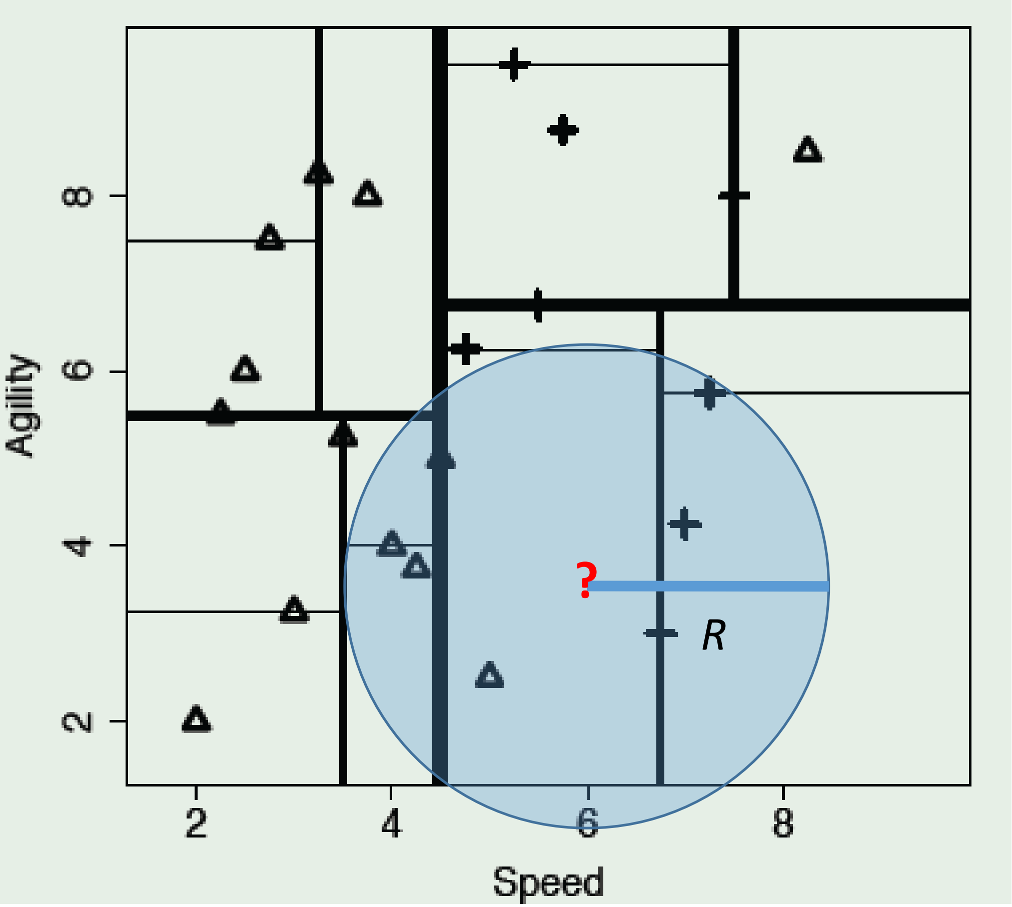
Finding the nearest neighbor by K-d tree

- 

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Finding the nearest neighbor by K-d tree

- K-NN by k-d tree
  - Let **t** be a ordered set with the capacity of *k*
- Given a radius *R*, finding the neighbors within a hypersphere $x_1{}^2 + x_2{}^2 + \cdots + x_k{}^2 = R$



Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Predicting Continuous Targets

- the average value in the neighborhood

$$\mathbf{M}_{k}(\mathbf{q}) = \frac{1}{k}\sum_{i=1}^{k} t_i$$

Predicting Continuous Targets

- Example:
  - A dataset of whiskeys listing the age (in years) and the rating (between 1 and 5, with 5 being the best) and the bottle price of each whiskey.

ID	Age	Rating	Price	ID	Age	Rating	Price
1	0	2	30.00	11	19	5	500.00
2	12	3.5	40.00	12	6	4.5	200.00
3	10	4	55.00	13	8	3.5	65.00
4	21	4.5	550.00	14	22	4	120.00
5	12	3	35.00	15	6	2	12.00
6	15	3.5	45.00	16	8	4.5	250.00
7	16	4	70.00	17	10	2	18.00
8	18	3	85.00	18	30	4.5	450.00
9	18	3.5	78.00	19	1	1	10.00
10	16	3	75.00	20	4	3	30.00

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Predicting Continuous Targets

- Normalized data:

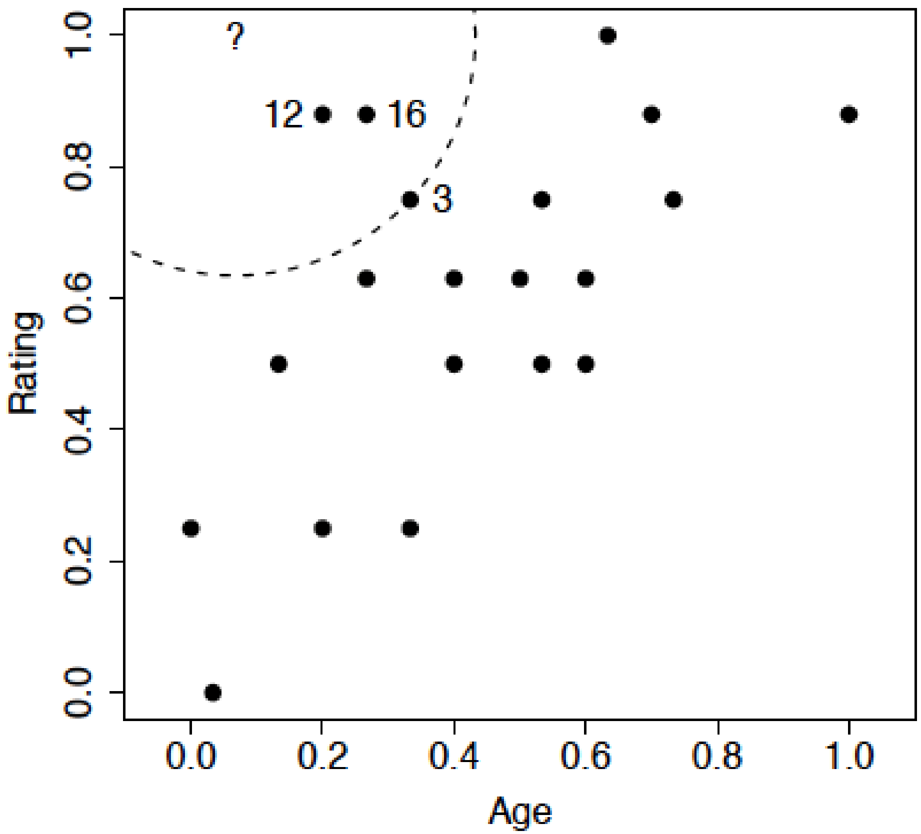
ID	Age	Rating	Price	ID	Age	Rating	Price
1	0.0000	0.25	30.00	11	0.6333	1.00	500.00
2	0.4000	0.63	40.00	12	0.2000	0.88	200.00
3	0.3333	0.75	55.00	13	0.2667	0.63	65.00
4	0.7000	0.88	550.00	14	0.7333	0.75	120.00
5	0.4000	0.50	35.00	15	0.2000	0.25	12.00
6	0.5000	0.63	45.00	16	0.2667	0.88	250.00
7	0.5333	0.75	70.00	17	0.3333	0.25	18.00
8	0.6000	0.50	85.00	18	1.0000	0.88	450.00
9	0.6000	0.63	78.00	19	0.0333	0.00	10.00
10	0.5333	0.50	75.00	20	0.1333	0.50	30.00

- Query: AGE = 0.0667 and RATING = 1.00
- K = 3

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Predicting Continuous Targets

- Result



$$\frac{200.00 + 250.00 + 55.00}{3} = 168.33$$

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Predicting Continuous Targets

- Weighted KNN for continuous targets

$$\mathbf{M}_k(\mathbf{q}) = \frac{\sum_{i=1}^{k} \frac{1}{d(\mathbf{q}, \mathbf{x}_i)^2} t_i}{\sum_{i=1}^{k} \frac{1}{d(\mathbf{q}, \mathbf{x}_i)^2}}$$

Similarity of binary features

- Example:
  - A binary dataset listing the behavior of two individuals on a website during a trial period and whether or not they subsequently signed-up for the website.

ID	Profile	FAQ	Help Forum	Newsletter	Liked	Signup
1	1	1	1	0	1	Yes
2	1	0	0	0	0	No

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Similarity of binary features

- Example:
  - A binary dataset listing the behavior of two individuals on a website during a trial period and whether or not they subsequently signed-up for the website.

ID	Profile	FAQ	Help Forum	Newsletter	Liked	Signup
1	1	1	1	0	1	Yes
2	1	0	0	0	0	No

  - signup = ? if **q** = (1, 0, 1, 0, 0)

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Similarity of binary features

- *CP*: co-presence <1, 1>
- *CA*: co-absence <0, 0>
- *PA*: presence-absence <1, 0>
- *AP*: absence-presence <0, 1>

		q Pres.	**q** Abs.
$\mathbf{d}_1$	Pres.	CP=2	PA=0
	Abs.	AP=2	CA=1

		q Pres.	**q** Abs.
$\mathbf{d}_2$	Pres.	CP=1	PA=1
	Abs.	AP=0	CA=3

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Similarity of binary features

- Russel-Rao similarity
  - The ratio between the number of CP and the total number of binary features considered.

$$S_{RR}(\mathbf{q}, \mathbf{x}) = \frac{CP(\mathbf{q}, \mathbf{x})}{|\mathbf{q}|}$$

  - where $|\mathbf{q}|$ is the total number of binary features

  - Example:

$$S_{RR}(\mathbf{q}, \mathbf{d_1}) = \frac{2}{5} = 0.4$$

$$S_{RR}(\mathbf{q}, \mathbf{d_2}) = \frac{1}{5} = 0.2$$

    - **q** is more similar to $\mathbf{d}_1$ then $\mathbf{d}_2$.

		q	
		Pres.	Abs.
$\mathbf{d}_1$	Pres.	CP=2	PA=0
	Abs.	AP=2	CA=1

		q	
		Pres.	Abs.
$\mathbf{d}_2$	Pres.	CP=1	PA=1
	Abs.	AP=0	CA=3

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Similarity of binary features

- Sokal-Michener similarity
  - the ratio between the total number of CP and CA, and the total number of binary features considered.

$$\boldsymbol{S_{SM}}(\mathbf{q},\mathbf{x}) = \frac{\boldsymbol{CP}(\mathbf{q},\mathbf{x}) + \boldsymbol{CA}(\mathbf{q},\mathbf{x})}{|\mathbf{q}|}$$

  - Example:

$$\boldsymbol{S_{SM}}(\mathbf{q},\mathbf{d_1}) = \frac{\mathbf{2+1}}{\mathbf{5}} = \mathbf{0.6}$$

$$\boldsymbol{S_{SM}}(\mathbf{q},\mathbf{d_2}) = \frac{\mathbf{1+3}}{\mathbf{5}} = \mathbf{0.8}$$

    - **q** is more similar to $\mathbf{d}_2$ then $\mathbf{d}_1$.

		q Pres.	**q** Abs.
$\mathbf{d}_1$	Pres.	CP=2	PA=0
	Abs.	AP=2	CA=1

		q Pres.	**q** Abs.
$\mathbf{d}_2$	Pres.	CP=1	PA=1
	Abs.	AP=0	CA=3

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

Similarity of binary features

- Jaccard similarity
  - ignoring CA

$$S_J(\mathbf{q},\mathbf{x}) = \frac{CP(\mathbf{q},\mathbf{x})}{CP(\mathbf{q},\mathbf{x}) + PA(\mathbf{q},\mathbf{x}) + AP(\mathbf{q},\mathbf{x})}$$

  - Example:

$$S_J(\mathbf{q},\mathbf{d}_1) = \frac{2}{2+0+2} = 0.5$$

$$S_J(\mathbf{q},\mathbf{d}_2) = \frac{1}{1+1+0} = 0.5$$

    - **q** is equally similar to $\mathbf{d}_1$ and $\mathbf{d}_2$.

		q Pres.	**q** Abs.
$\mathbf{d}_1$	Pres.	CP=2	PA=0
	Abs.	AP=2	CA=1

		q Pres.	**q** Abs.
$\mathbf{d}_2$	Pres.	CP=1	PA=1
	Abs.	AP=0	CA=3

Figuring by: John D. Kelleher, et al, "Fundamentals of Machine Learning for Predictive Data Analytics - Algorithms, Worked Examples, and Case Studies," *MIT Press*, 2015.

KD Tree in sklearn

- An example of querying for 3-nearest neighbors

```
import numpy as np
from sklearn.neighbors import KDTree

np.random.seed(0)
X = np.random.random((10, 3))  # 10 points in 3 dimensions
tree = KDTree(X, leaf_size=2)
dist, ind = tree.query([X[0]], k = 3)
print(ind)                          # indices of 3 closest neighbors
print(dist)
```

KD Tree in sklearn

- An example of querying for neighbors within a given radius

```
import numpy as np
from sklearn.neighbors import KDTree

np.random.seed(0)
X = np.random.random((10, 3))  # 10 points in 3 dimensions
tree = KDTree(X, leaf_size=2)
print(tree.query_radius([X[0]], r=0.3, count_only=True))
ind, dist = tree.query_radius( [X[0]],
                               r=0.3,
                               count_only = False,
                               return_distance = True)
print(ind)
print(dist)
```

KD Tree and Curse of Dimensionality

- KD tree suffers from curse of dimensionality
  - The number of features is high → high dimension
  - The KD tree becomes higher
  - The search time also be increased

Ball Tree

- A ball tree is a space partitioning data structure for organizing points in a multi-dimensional space.
- It partitions data points into a nested set of **hyperspheres** known as "balls".
- It is useful for nearest neighbor search.
- References:
  - Omohundro, Stephen M. "Five Balltree Construction Algorithms", 1989.
  - M. Dolatshah, et al, "Ball*-tree: Efficient spatial indexing for constrained nearest-neighbor search in metric spaces," *arXiv*, Nov. 2015.

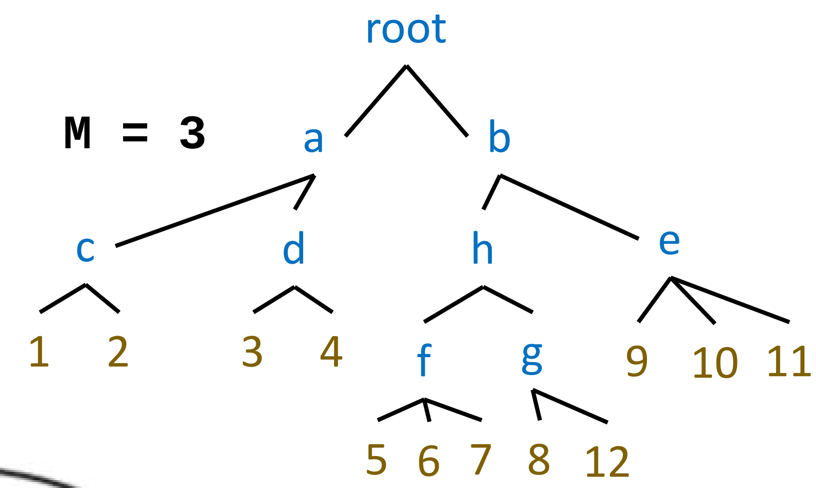
Ball Tree

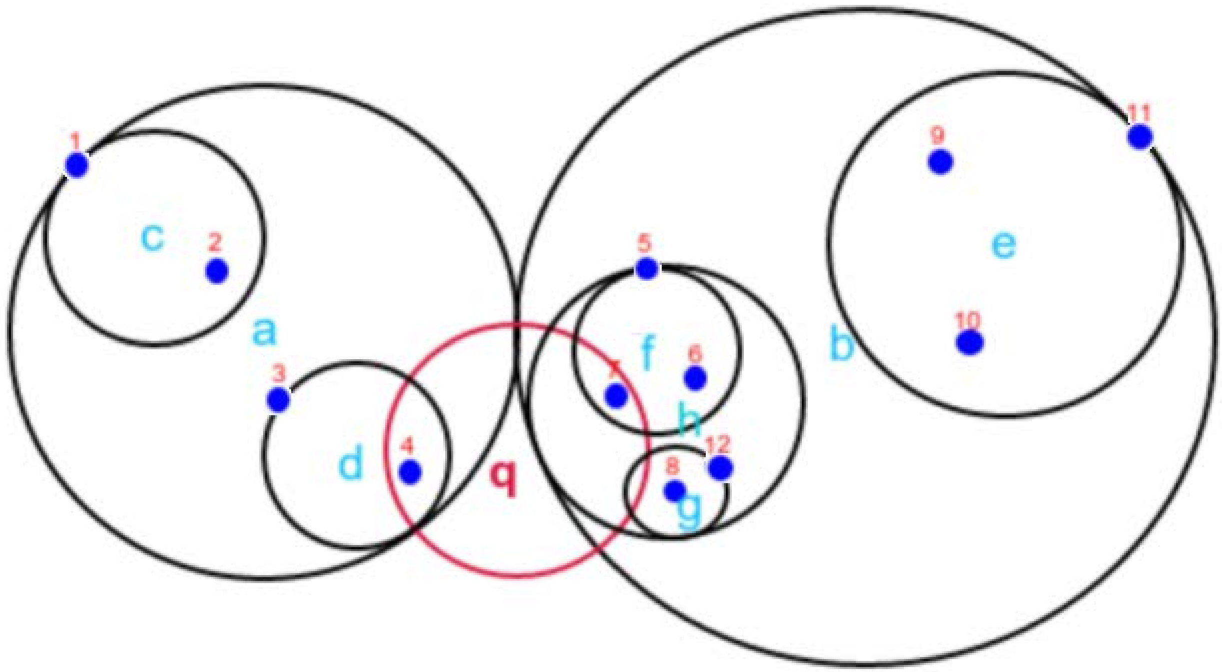
- Construction Algorithm

```
function construct_balltree is
      input:
          D, an array of data points
      output:
          B, the root of a constructed ball tree
      if M points remains then
          create a leaf B containing the M points in D
          return B
      else
          let c be the dimension of greatest spread(PCA)
          let p be the central point selected considering c
          let L,R be the sets of points lying to the left and
              right of the median along dimension c
          create B with two children:
              B.pivot = p
              B.child1 = construct_balltree(L),
              B.child2 = construct_balltree(R),
              let B.radius be maximum distance from p among children
          return B
      end if
  end function
```

Ball Tree

- Example: 12 data instances





Ball Tree

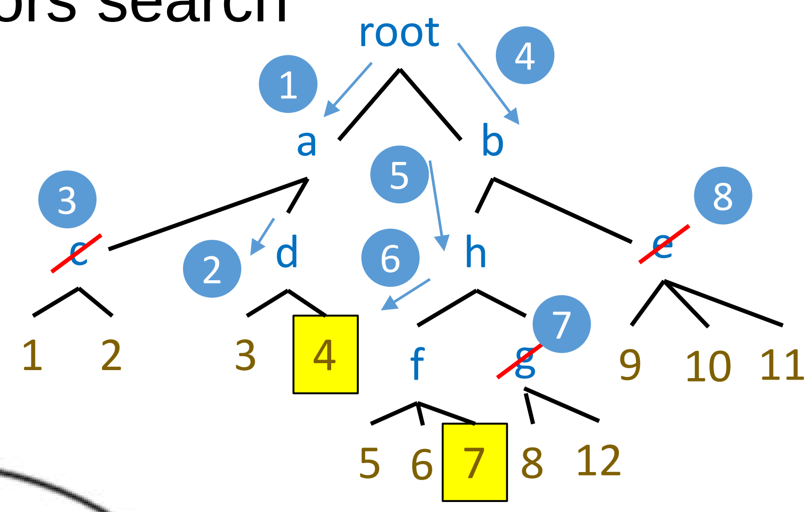
- Nearest neighbor search

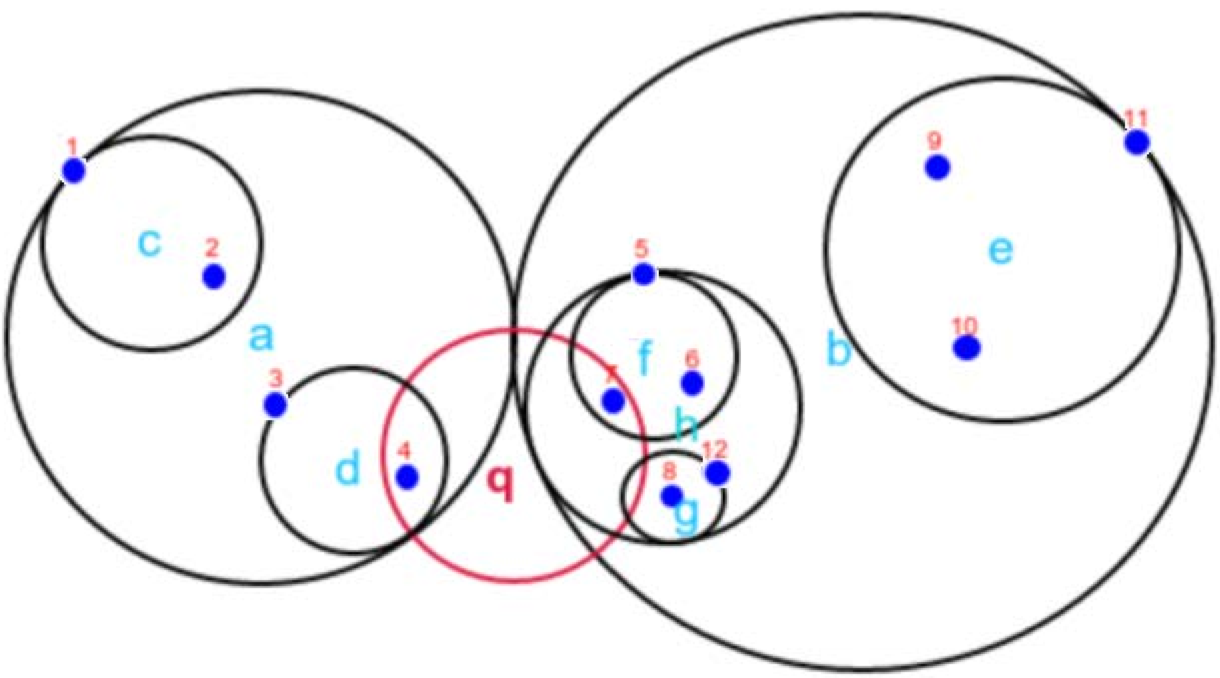
```
function knn_search is
      input:
          t, the target point for the query
          k, the number of nearest neighbors of t to search for
          Q, max-first priority queue containing at most k points
          B, a node, or ball, in the tree
      output:
          Q, containing the k nearest neighbors from within B
      if distance(t, B.pivot) - B.radius ≥ distance(t, Q.first) then
          return Q unchanged
      else if B is a leaf node then
          for each point p in B do
              if distance(t, p) < distance(t, Q.first) then
                  add p to Q
                  if size(Q) > k then
                      remove the furthest neighbor from Q
      else
          let child1 be the child node closest to t
          let child2 be the child node furthest from t
          knn_search(t, k, Q, child1)
          knn_search(t, k, Q, child2)
```

∞ if Q is empty

Ball Tree

- Example: Two nearest neighbors search





Linear Algebra: Matrix

- Each data instance has *n* features
  - $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$,
  - $\mathbf{x}^\mathrm{T} = [x_1 \quad x_2 \quad \cdots \quad x_n]$
- *m* data instances➔ $m \times n$ matrix
  - $\boldsymbol{A} = \begin{bmatrix} \mathbf{x_1}^\mathrm{T} \\ \mathbf{x_2}^\mathrm{T} \\ \vdots \\ \mathbf{x}_{\boldsymbol{m}}{}^\mathrm{T} \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & & \ddots & \\ x_{m1} & x_{m2} & & x_{mn} \end{bmatrix}$

Linear Algebra: Matrix

- M is a symmetric matrix if $M^{\mathrm{T}} = M$
  - $A^{\mathrm{T}}A$ is a symmetric matrix
    - Because $(A^{\mathrm{T}}A)^{\mathrm{T}} = A^{\mathrm{T}}A$

- U is an unitary matrix if $U^*U = UU^* = I$
  - where I is the identity matrix
  - *: conjugate transpose
    - if $\boldsymbol{A} = \begin{bmatrix} 1 & -2-3i \\ 1+4i & 5i \end{bmatrix}, \boldsymbol{A}^* = \begin{bmatrix} 1 & 1-4i \\ -2+3i & -5i \end{bmatrix}$

- Q is an orthogonal matrix if $Q^{\mathrm{T}}Q = QQ^{\mathrm{T}} = I$ and Q is a square matrix with real entries

Linear Algebra: Matrix

- $\boldsymbol{A}$ is an $m \times n$ matrix
  - Column space: $\boldsymbol{A}\mathbf{x}$, where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$
    - a space of $m \times 1$ matrices
  - Row space: $\mathbf{x}\boldsymbol{A}$, where $\mathbf{x} = [x_1, x_2, \ldots, x_m]$
    - a space of $1 \times n$ matrices

Linear Algebra: Eigenvalues & Eigenvetcors

- A is an $m \times n$ matrix
  - An eigenvector of A is a non-zero vector $\mathbf{v}$ such that

$$A\mathbf{v} = \lambda\mathbf{v}$$

or

$$\mathbf{v}^{\mathrm{T}}A = \lambda\mathbf{v}^{\mathrm{T}}$$

  - The scalar λ is the eigenvalue corresponding to $\mathbf{v}$
  - Given a non-zero real number s, $s\mathbf{v}$ and $\mathbf{v}$ have the same eigenvalue
    - $A\mathbf{v} = \lambda\mathbf{v}$
    - $A(s\mathbf{v}) = \lambda(s\mathbf{v})$
  - We often let every eigenvector is normalized (the length is one)

Linear Algebra: Eigenvalues & Eigenvetcors

- Linearly independent
  - A set of n vectors, $V = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$
  - V is linearly independent if $a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \ldots + a_n\mathbf{v}_n = \text{zero vector}$, can only be satisfied by all a_i are zeros

- Suppose that A has n linearly independent eigenvectors,
  - $\boldsymbol{A}$ is an $m \times n$ matrix
  - Q is an $n \times n$ matrix $= [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n]$
  - $\boldsymbol{\Lambda}$ is an $n \times n$ diagonal matrix with n eigenvalue
  - $AQ = Q\,\boldsymbol{\Lambda}$

Linear Algebra: Eigenvalues & Eigenvetcors

- Every **real symmetric matrix** can be decomposed as follows
  - $\boldsymbol{A}$ is an $n \times n$ real symmetric matrix
  - $A = Q\Lambda Q^{\mathrm{T}}$
    - Q contains n orthogonal eigenvectors
      - $QQ^{\mathrm{T}} = I$
    - Λ is an $n \times n$ diagonal matrix contains n eigenvalues

Linear Algebra: Eigenvalues & Eigenvetcors

- Singular value decomposition, SVD
  - Every real matrix has a singular value decomposition
    - A is an $m \times n$ matrix
    - $A = UDV^{\mathrm{T}}$
      - U is an $m \times m$ matrix, $UU^{\mathrm{T}} = U^{\mathrm{T}}U = I$
      - D is an $m \times n$ diagonal matrix
      - V is an $n \times n$ matrix, $VV^{\mathrm{T}} = V^{\mathrm{T}}V = I$

  - But the same is not true of the eigenvalue decomposition.
    - For example, if a matrix is not square, the eigen decomposition is not defined, and we must use a singular value decomposition instead.

Linear Algebra: Eigenvalues & Eigenvetcors

- $\boldsymbol{A}$ is an $m \times n$ real matrix
  - $A = UDV^{\mathrm{T}}$
- $A^{\mathrm{T}}A$ is an $n \times n$ real symmetric matrix
  - $A^{\mathrm{T}}A = (UDV^{\mathrm{T}})^{\mathrm{T}}UDV^{\mathrm{T}} = VDU^{\mathrm{T}}UDV^{\mathrm{T}} = VD^2V^{\mathrm{T}}$
  - $Q = V$
  - $\Lambda = D^2$
- AA^{T} has the same property

Linear Algebra: Eigenvalues & Eigenvetcors

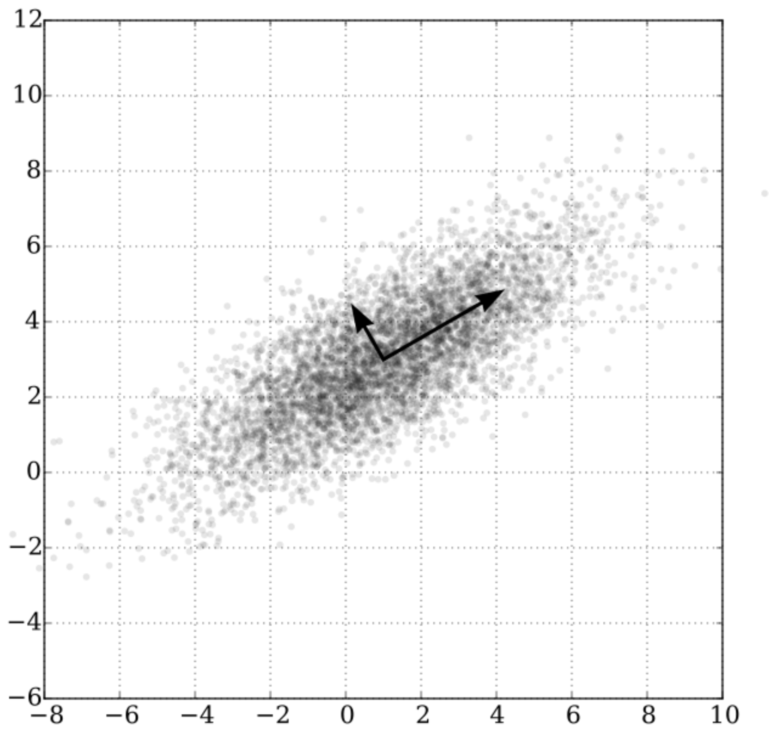
- The center of *m* data instances
  - $\bar{\mathbf{x}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i$
- *m* data instances➔ $m \times n$ matrix
  - *m* data instances
  - *n* dimensions (features)

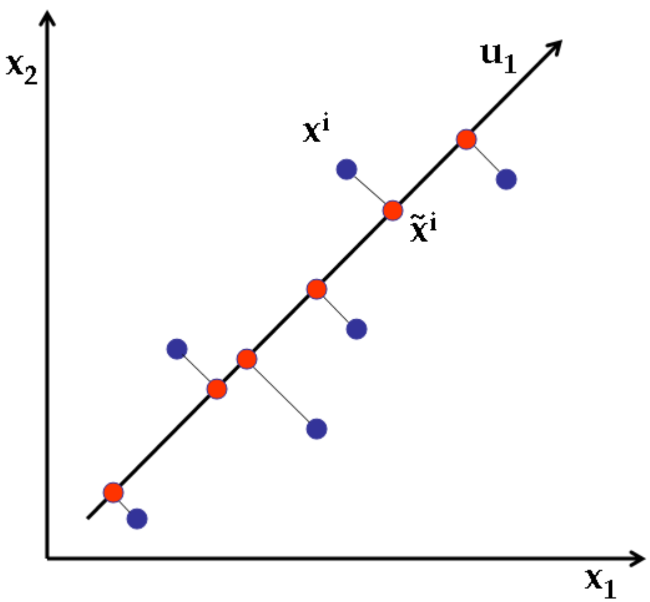
  - $A = \begin{bmatrix} (\mathbf{x}_1 - \bar{\mathbf{x}})^{\mathrm{T}} \\ (\mathbf{x}_2 - \bar{\mathbf{x}})^{\mathrm{T}} \\ \vdots \\ (\mathbf{x}_m - \bar{\mathbf{x}})^{\mathrm{T}} \end{bmatrix}$

  - $A^{\mathrm{T}}A$ ($n \times n$) is the covariance matrix to describe the data variance of all dimensions (features)

PCA

- Principal component analysis
  - The main linear technique for **dimensionality reduction**.
  - It performs a linear mapping of the data to a **lower-dimensional space** in such a way that the variance of the data in the low-dimensional representation is maximized.





Figuring by: Wikipedia, https://en.wikipedia.org/wiki/Principal_component_analysis

PCA

- Principal components:
  - The eigenvectors ordered with eigenvalues
  - $A^{\mathrm{T}}A = Q\Lambda Q^{\mathrm{T}}$
    - where $\Lambda = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_n \end{bmatrix}, \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$
    - $Q = [\mathbf{v_1} \quad \mathbf{v_2} \quad \cdots \quad \mathbf{v_n}]$
  - The data can be represented by in a space with the basis of Q
    - $\mathbf{y}^{\mathrm{T}} = (\mathbf{x} - \bar{\mathbf{x}})^{\mathrm{T}} Q$
    - The axis $\mathbf{v_1}$ is more importance than $\mathbf{v_2}$
    - Choose $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}$, where λ_k larger than a threshold, we call these axes are principal components

PCA

- PCA in sklearn

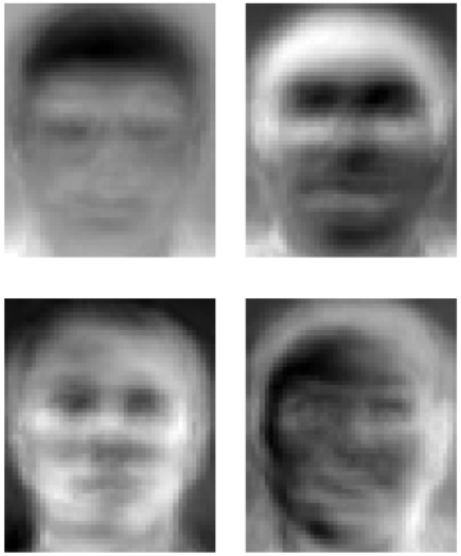
```
import numpy as np
from sklearn import datasets
from sklearn import decomposition

np.random.seed(5)
iris = datasets.load_iris()
X = iris.data
y = iris.target

pca = decomposition.PCA(n_components=3)
pca.fit(X)
print( pca.components_ )
    #[[ 0.36158968 -0.08226889  0.85657211  0.35884393]
    # [ 0.65653988  0.72971237 -0.1757674  -0.07470647]
    # [-0.58099728  0.59641809  0.07252408  0.54906091]]
```

PCA Example: Face Recognition

- L. Sirovich; M. Kirby (1987). "Low-dimensional procedure for the characterization of human faces". *Journal of the Optical Society of America A*. 4 (3): 519–524.
- Eigenfaces
  - Eigenvectors of a training set of *m* face images
  - Each image: $r \times c$ pixels ➔ let $n = r \times c$ ➔a vector of *n* intensities
  - The average of *m* face images➔$\bar{\mathbf{x}}$
  - *m* face images - $\bar{\mathbf{x}}$ ➔ $A = m \times n$ matrix
  - Finding the eigenvectors of $A^{\mathrm{T}}A$
  - Each eigenvector consists of *n* intensities
  - eigenvectors ➔ eigenfaces



AT&T Laboratories Cambridge

PCA Example: Face Recognition

- Choose the principal components

$$\bar{\lambda} = (\lambda_1 + \lambda_n + \cdots + \lambda_n)$$

  - Given a threshold *t*, we choose *k* principal components such that

$$\frac{(\lambda_1+\lambda_n+\cdots+\lambda_k)}{\bar{\lambda}} > t$$

  - Therefore, each training face image can be represented by
    - $\mathbf{y}^{\mathrm{T}} = (\mathbf{x} - \bar{\mathbf{x}})^{\mathrm{T}} Q$
    - $\Rightarrow \mathbf{y}^{\mathbf{T}} Q^{\mathbf{T}} = (\mathbf{x} - \bar{\mathbf{x}})^{\mathrm{T}}$
    - $\Rightarrow \mathbf{x} = Q\boldsymbol{y} + \bar{\mathbf{x}}$
    - $y_1 \times$ 1st eigenface + $y_2 \times$ 2nd eigenface + … + $y_k \times$ kth eigenface

PCA Example: Face Recognition

- Face recognition
  - input face: $\mathbf{z}$
  - $\mathbf{w}^{\mathbf{T}} = (\mathbf{z} - \bar{\mathbf{x}})^{\mathrm{T}} Q$
  - Find the nearest neighbour of $\mathbf{w}^{\mathbf{T}}$ from the transformed training set (all $\mathbf{y}$)

PCA Example: Feature Analysis

- Reference:
  - Jeff Jauregui "Principal component analysis with linear algebra", 2012.
- Sibley's Bird Database of North American birds
- 100 bird species. ➔ $m = 100$
- In studying **the size of a bird**, each specie should be observed by three features: **length**, **wingspan**, and **weight**. ➔ $n = 3$



PCA Example: Feature Analysis

- $A^{\mathrm{T}}A = \begin{bmatrix} 91.43 & 171.92 & 297.99 \\ 545.21 & 373.92 & 545.21 \\ 297.99 & 171.92 & 1297.26 \end{bmatrix}$

- $\Rightarrow \lambda_1 = 1626.52, \lambda_2 = 128.99, \lambda_3 = 7.10$

- $\Rightarrow \mathbf{v}_1 = \begin{bmatrix} 0.22 \\ 0.41 \\ 0.88 \end{bmatrix}, \mathbf{v_2} = \begin{bmatrix} 0.25 \\ 0.85 \\ -0.46 \end{bmatrix}, \mathbf{v_3} = \begin{bmatrix} 0.97 \\ -0.32 \\ -0.08 \end{bmatrix}$
- Which of the feature is most significant to distinguish Sibley birds?

PCA Example: Feature Analysis

- The third entry, weight, of $\mathbf{v}_1$ is the largest
  - Weight is the most significant.
  - Wingspan is the next most important factor.
- $\mathbf{v}_2$ is also telling us something
  - Some birds are large size with small wingspan and large weight ➔ stoutness birds