Instance based learning

Master in Data Science and Advanced Analytics
BA and DS

Roberto Henriques

Instance-based learning

- Basic idea: “*When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck.*”

- Simplest form of learning
  - Training instances are searched to find instance that most closely resembles new instance
  - The instances themselves represent the knowledge

- Instance-based learning is lazy learning (vs eager learning)

Instance-based learning

⚠ Important concepts:

- feature space
- similarity measures

✎ Basic algorithm: nearest-neighbor

✎ Extensions: k-nearest neighbor (knn)

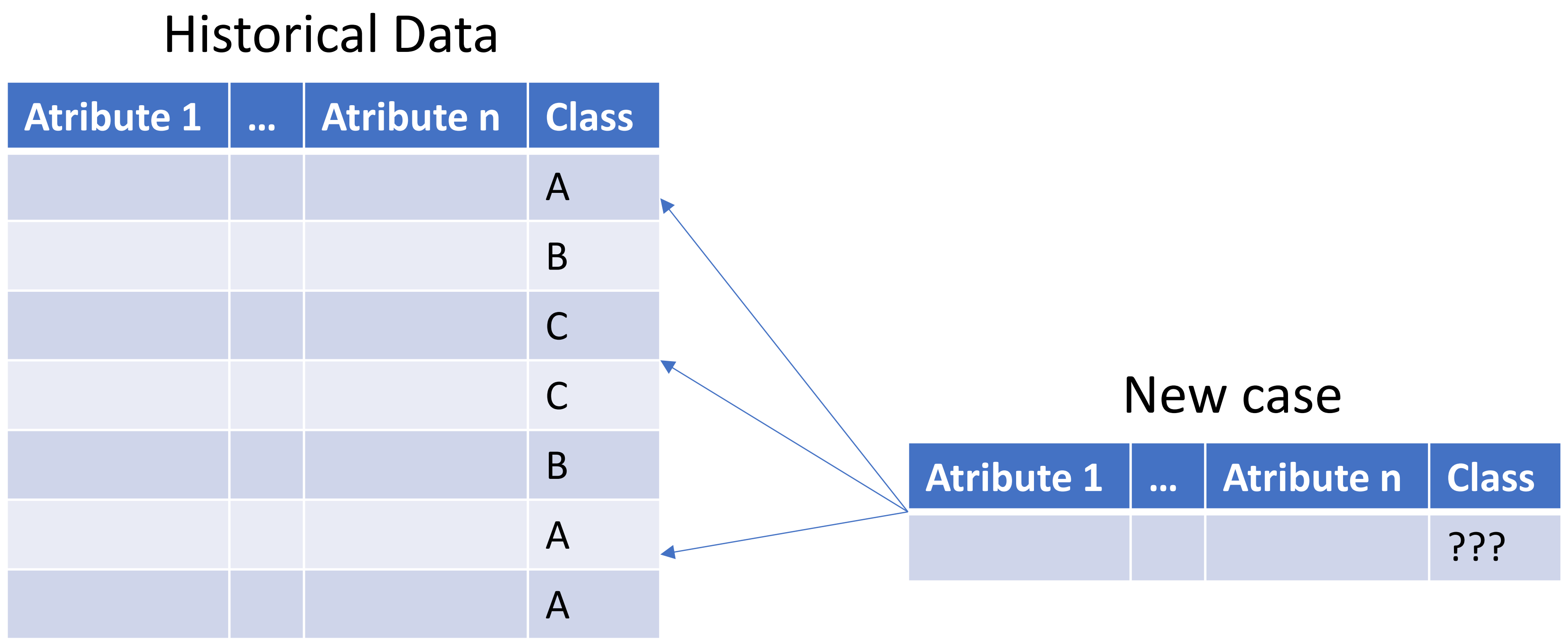
k-d trees

continuous targets

…

Instance-Based Classifiers

1) Store the training records
2) Use training records to predict the class label of unseen cases



Historical Data

Atribute 1	…	Atribute n	Class
			A
			B
			C
			C
			B
			A
			A

New case

Atribute 1	…	Atribute n	Class
			???

Instance Based Classifiers

Rote-learner: memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly

What if there is no exact match?

✓Use the “closest” point (nearest neighbor) to perform classification

Nearest Neighbor Classifier

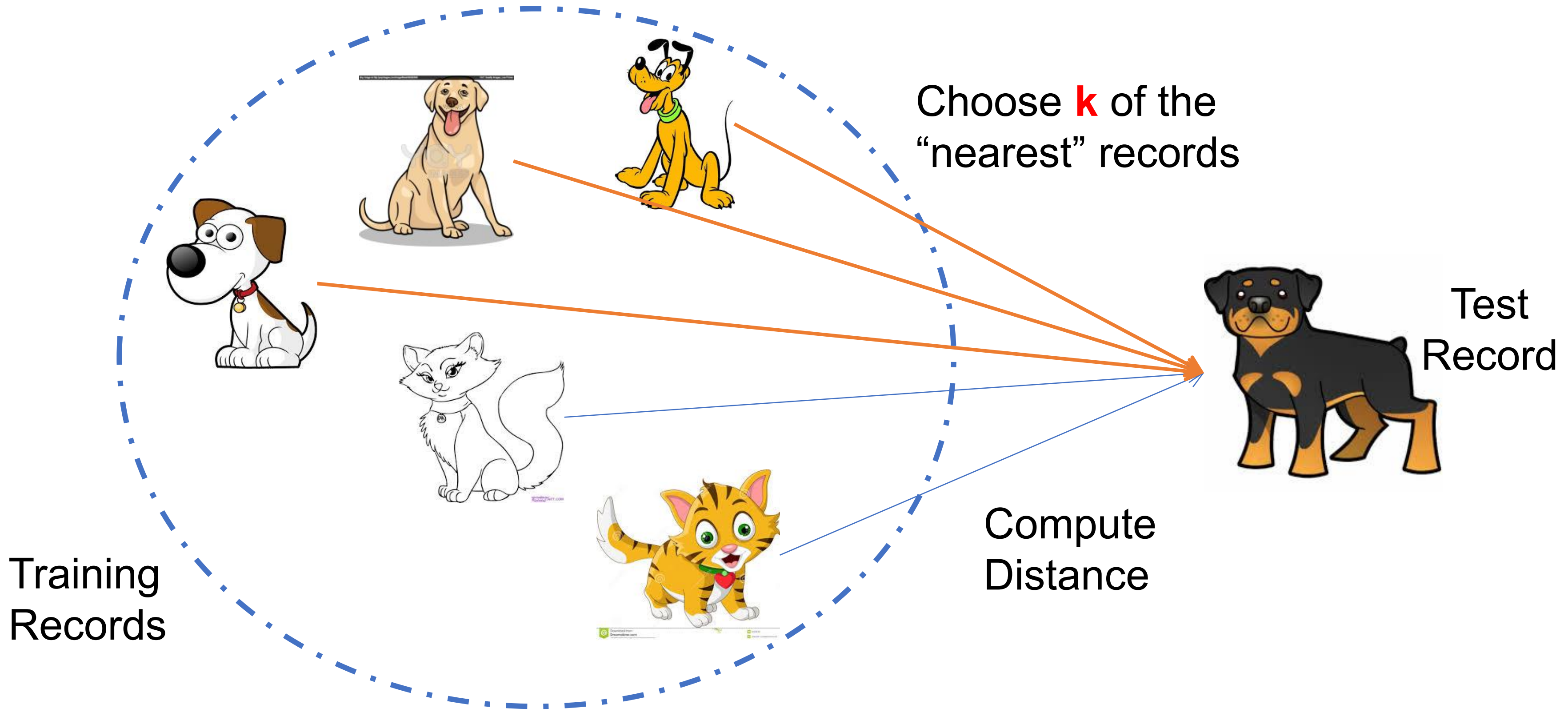
⚠ If we only use the closest neighbor outliers and mislabeled datapoints will have a big impact on our predictions!

Dilute the dependency of the algorithm on individual (possibly noisy) instances:

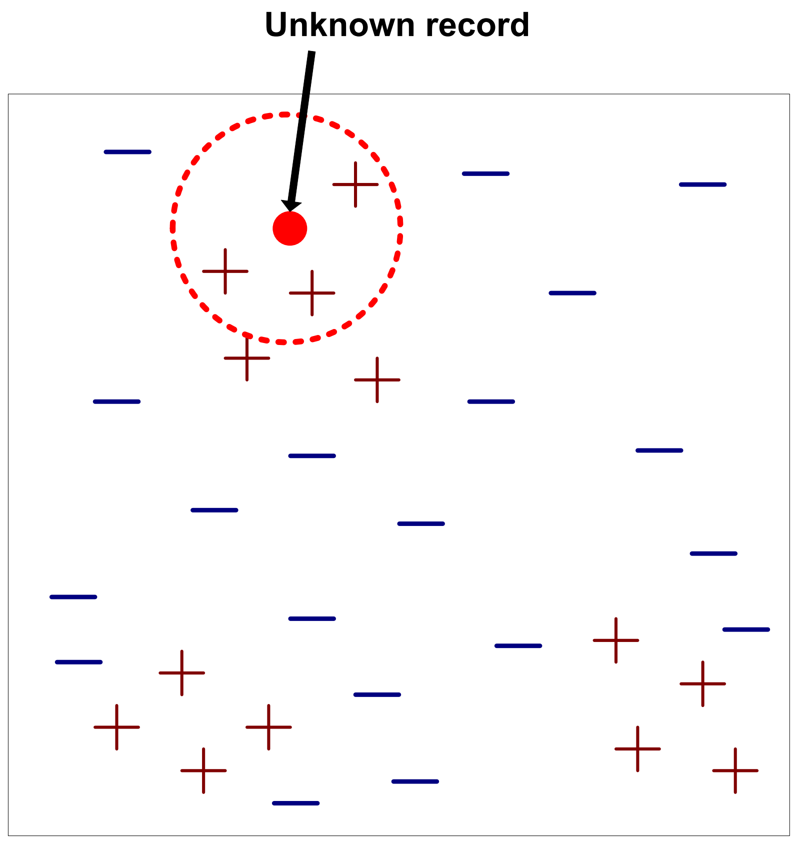
✓ Use the k "closest" points (k-nearest neighbors) to perform classification ⇨ KNN

KNN

- Basic idea: If it walks like a dog, barks like a dog, then it’s probably a dog



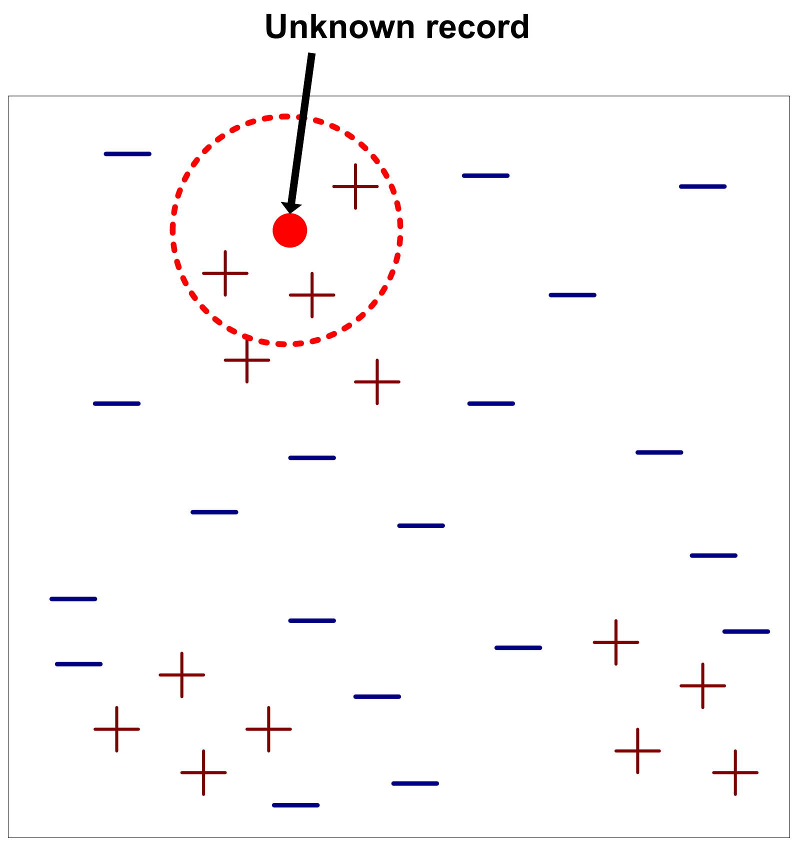
KNN



Requires three things:

1) The set of stored records
2) Distance metric to compute distance between records
3) The number of nearest neighbors to retrieve (value of *k*)

KNN

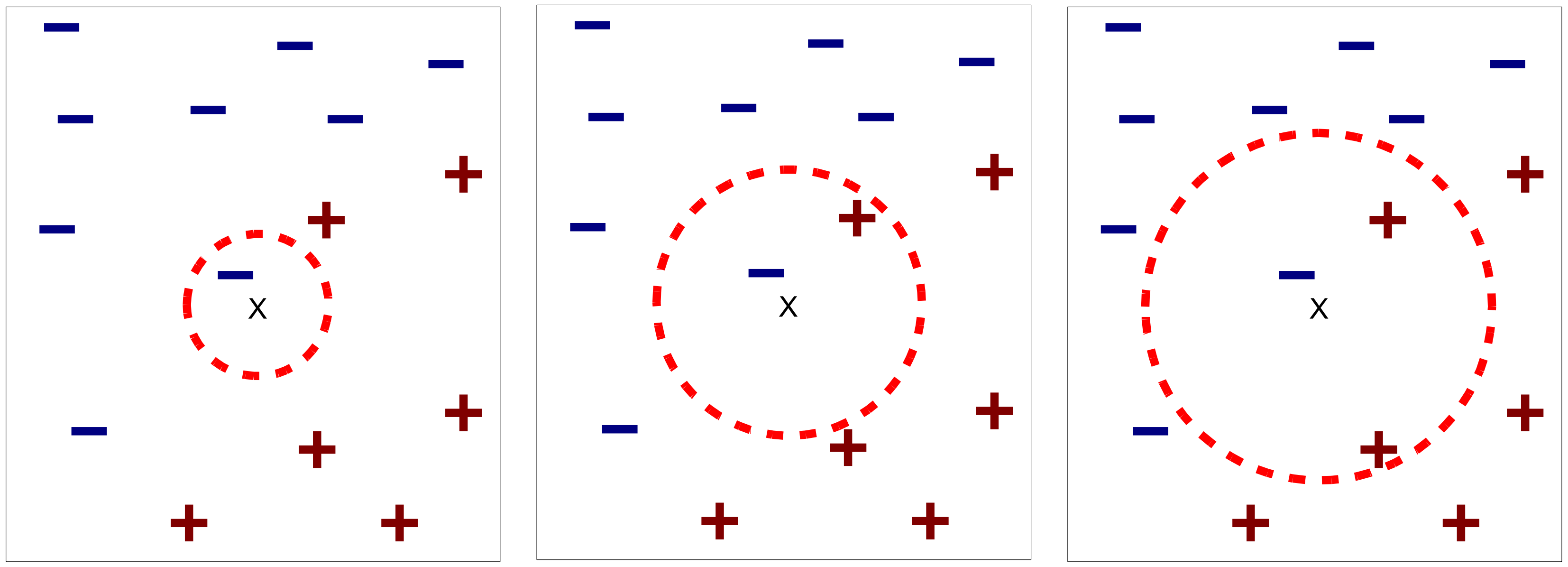


To classify an unknown record:

- ✓ Compute distance to other training records
- ✓ Identify *k* nearest neighbors
- ✓ Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

KNN

K-nearest neighbors of a record x are the k data points that have the smallest distance to x.

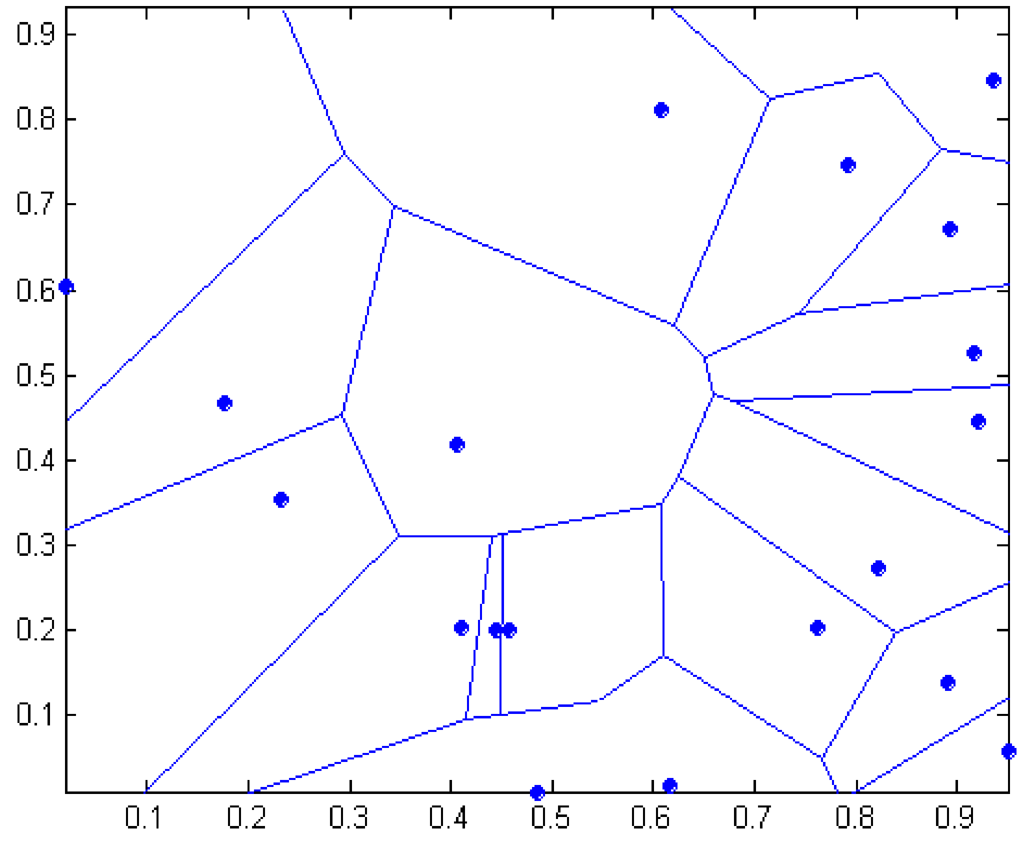


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

1 nearest-neighbor

Voronoi Diagram

Decision Boundary



Distance Metrics

In order to find the k closest neighbors we need to compute distance between two points.

Euclidean distance

$$d(x,y) = \sqrt{\sum_{k=1}^{p} (x_k - y_k)^2}$$

Manhattan distance

$$d(x,y) = \sum_{k=1}^{p} |x_k - y_k|$$

Minkowski distance

$$d(x,y) = \left(\sum_{k=1}^{p} |x_k - y_k|^r \right)^{\frac{1}{r}}$$

Determine the class

Standard approach: take the majority vote of class labels among the k-nearest neighbors

Alternative: Weigh the vote according to distance using the reciprocal of squared distance $\Rightarrow$

$$\boxed{w = 1/d^2}$$

$$\Downarrow$$

<u>closer neighbors have more weight in the vote</u>

Variant of KNN: Distance-Weighted KNN

- Use Distance-Weighted KNN to weight nearer neighbours more heavily

$$f(\mathbf{x}_q) := \frac{\sum_{i=1}^{k} w_i f(\mathbf{x}_i)}{\sum_{i=1}^{k} w_i} \quad \text{where } w_i = \frac{1}{d(\mathbf{x}_q, \mathbf{x}_i)^2}$$

- Now it can make sense to use *all training examples* instead of just k (Stepard’s method)

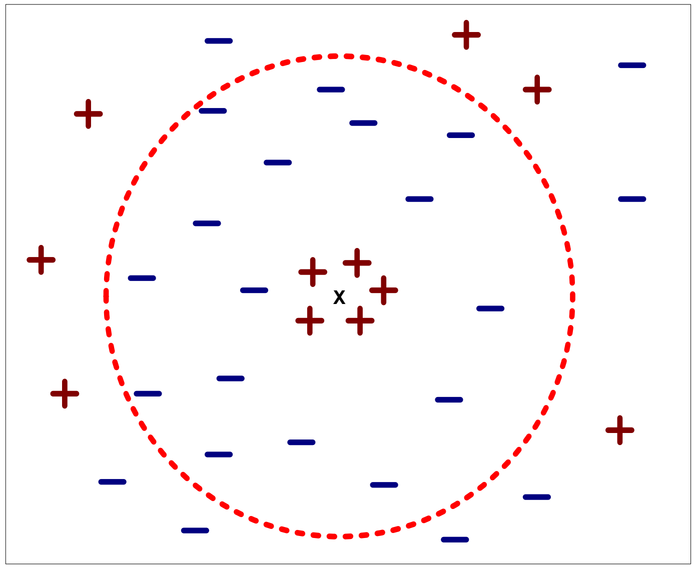
Choosing the value of k

k too small

- ⚠ sensitive to noise points
- ⚠ risk of overfitting

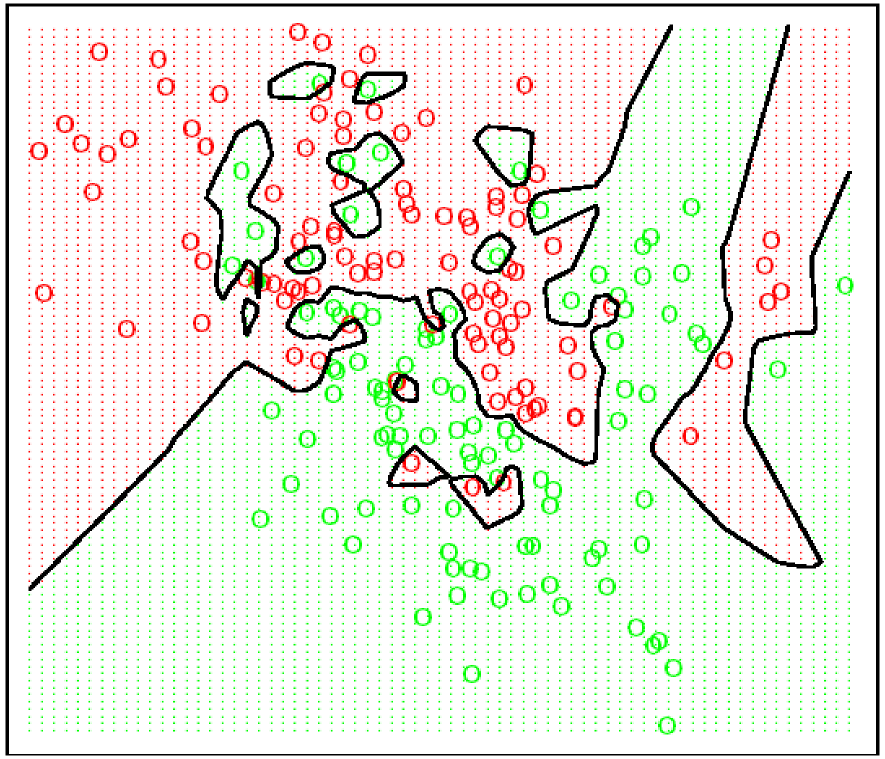
k too large

- ⚠ might not find the true pattern of the data
- ⚠ risk of underfitting

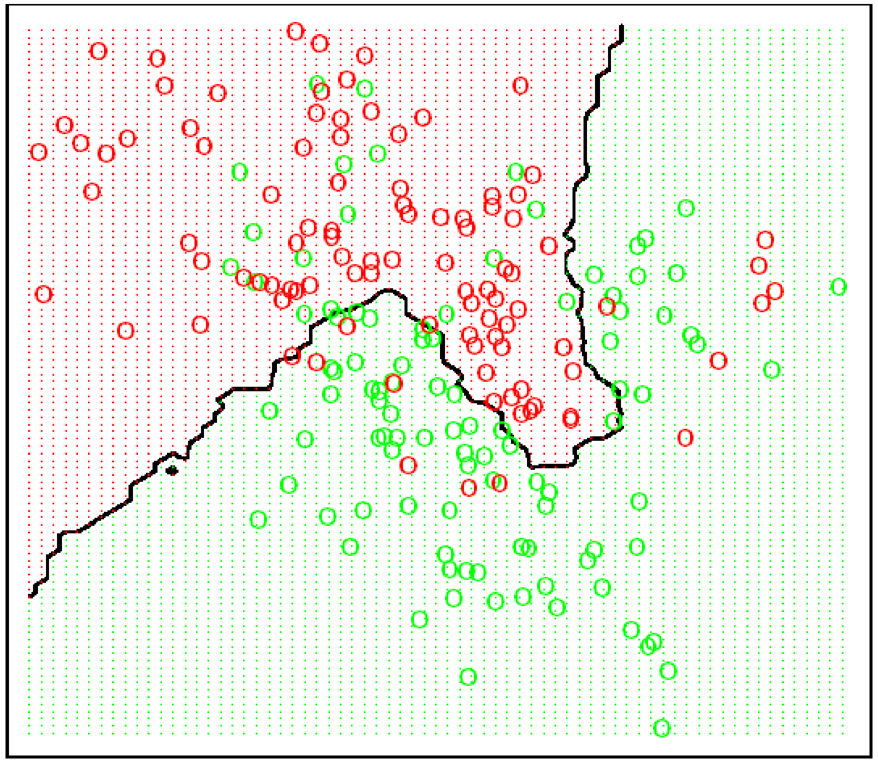


KNN frontiers

1-nn



15-nn



⚠ Small k

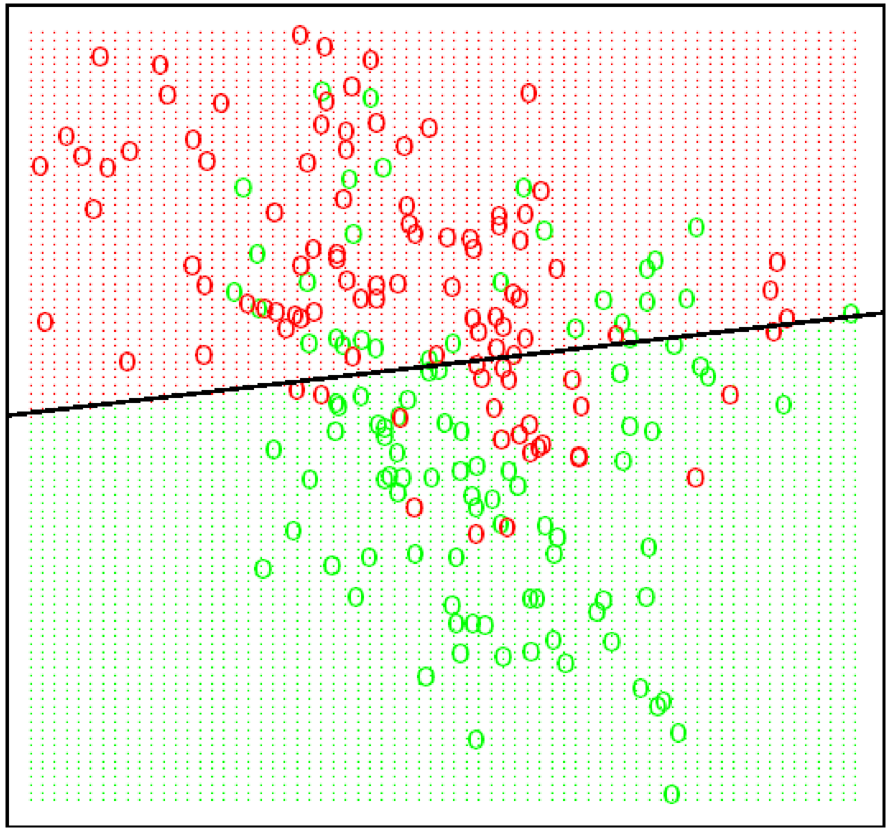
- ✓ Very sensitive to outliers
- ✓ Crisp frontiers

⚠ Large k

- ✓ Smooth frontiers
- ✓ Unable to detect small variations

If $k \to n$

- ⚠ for a balanced dataset the decision boundary will resemble a linear regression



Difficulties with KNN algorithms

Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes.

Examples:

- ✓ height of a person may vary from 1.5m to 2m
- ✓ weight of a person may vary from 50Kg to 90Kg
- ✓ income of a person may vary from €5k to €500k

Difficulties with KNN algorithms

- ⚠ There may be irrelevant attributes amongst the descriptive features (curse of dimensionality)   importance of feature selection

- ⚠ Calculating the distance of the test case to all training cases:
  - *requires a lot of memory to store the training set*
  - *requires a lot of time at the classification stage*

- ⚠ Very sensitive to outliers
- ⚠ Very sensitive to the distance function chosen

K-d Tree

Computing all distances in a large dataset might be prohibitive!

Solution: create an index of the instances that enables efficient retrieval of the nearest neighbor (efficient memory search)

K-d Tree

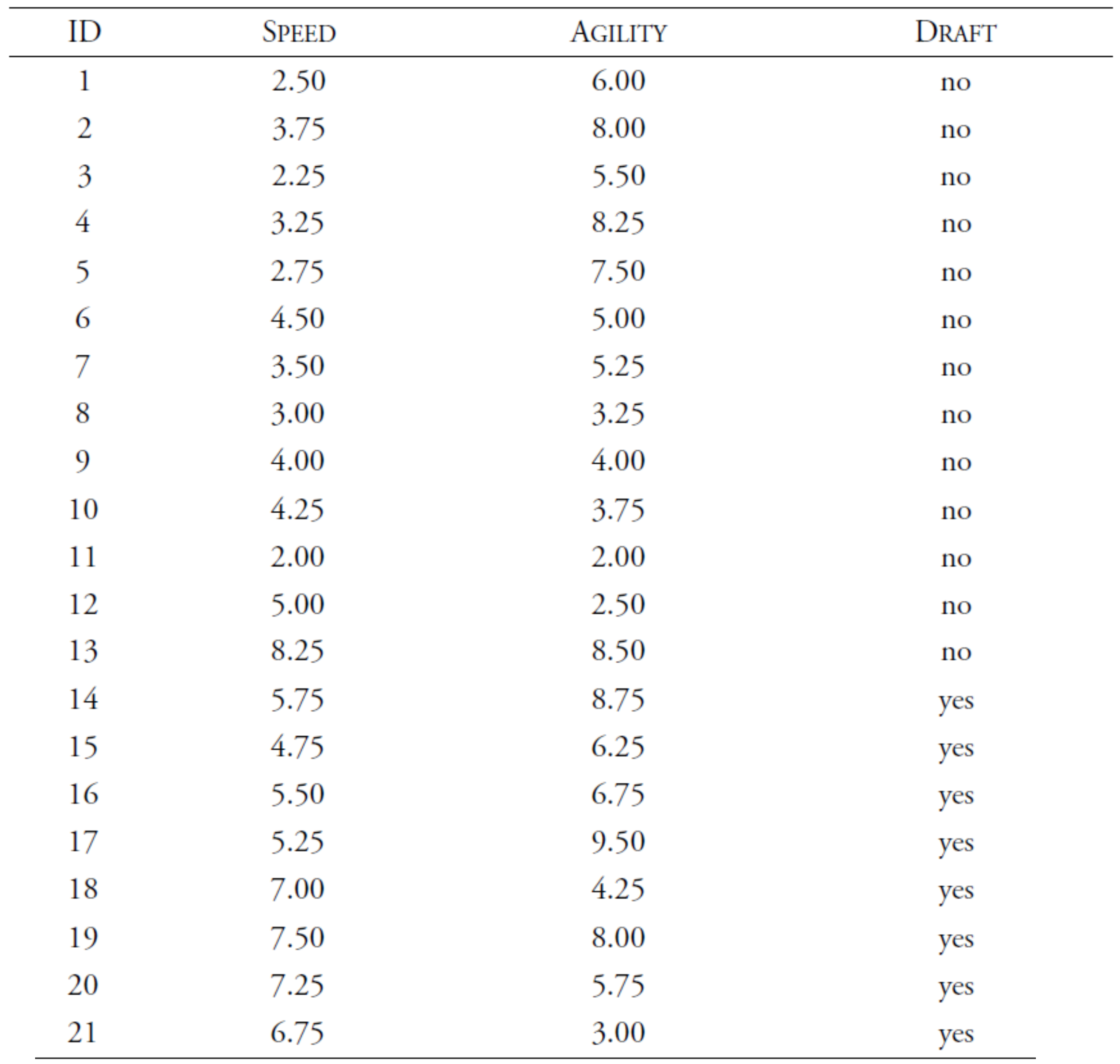
- One of the best known of those indexes is the k-d tree (k-dimensional tree)!
- Others: locality sensitivity hashing, r-trees, b-trees, m-trees,…

The k-d tree is a balanced binary tree in which each node (including leaf nodes) indexes one of the instances of the training set.

K-d Tree

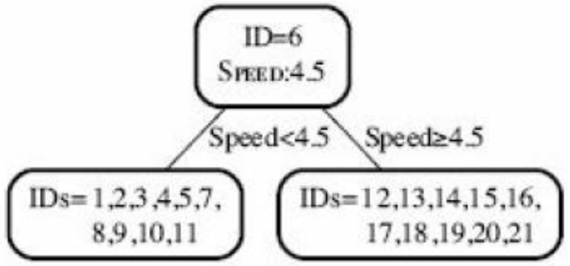
- Steps:
  1) Make a list of the descriptive features in random order
  2) Start with first feature of the list, find the median for it and split the data accordingly (the datapoint corresponding to the median will be the indexed point of that node)
  3) Do these splits until there are less than two datapoints in each node (these will be the leaf nodes)

K-d Tree

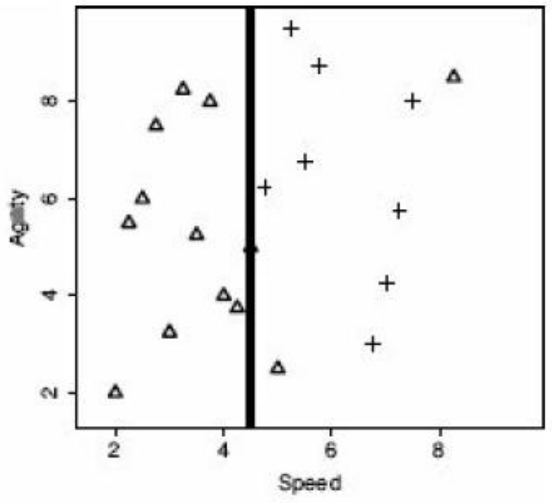


ID	SPEED	AGILITY	DRAFT
1	2.50	6.00	no
2	3.75	8.00	no
3	2.25	5.50	no
4	3.25	8.25	no
5	2.75	7.50	no
6	4.50	5.00	no
7	3.50	5.25	no
8	3.00	3.25	no
9	4.00	4.00	no
10	4.25	3.75	no
11	2.00	2.00	no
12	5.00	2.50	no
13	8.25	8.50	no
14	5.75	8.75	yes
15	4.75	6.25	yes
16	5.50	6.75	yes
17	5.25	9.50	yes
18	7.00	4.25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

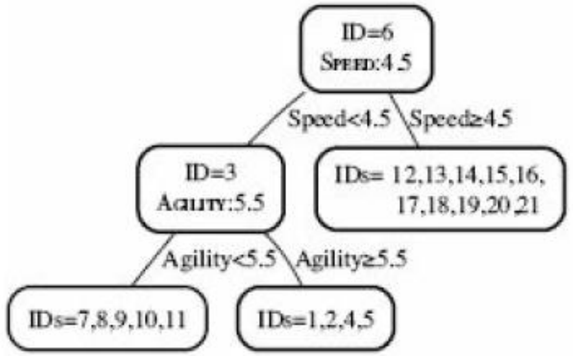
K-d Tree

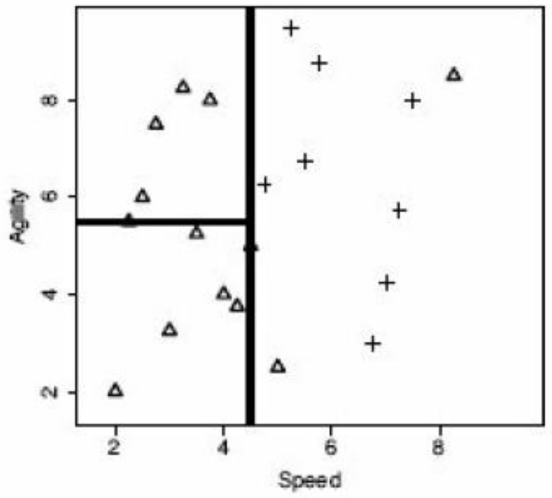

ID=6
SPEED:4.5
Speed<4.5
Speed≥4.5
IDs= 1,2,3,4,5,7, 8,9,10,11
IDs= 12,13,14,15,16, 17,18,19,20,21

(a)

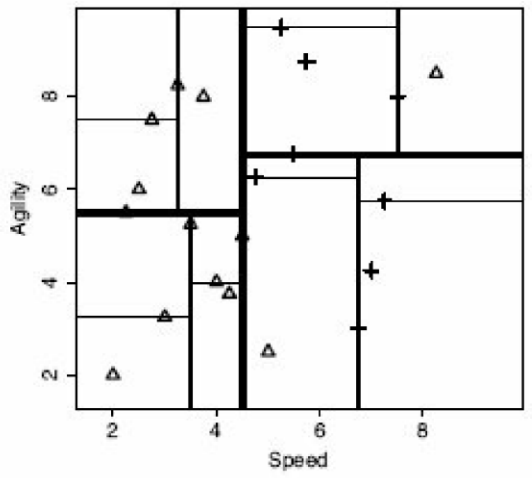

Agility
Speed
2
4
6
8

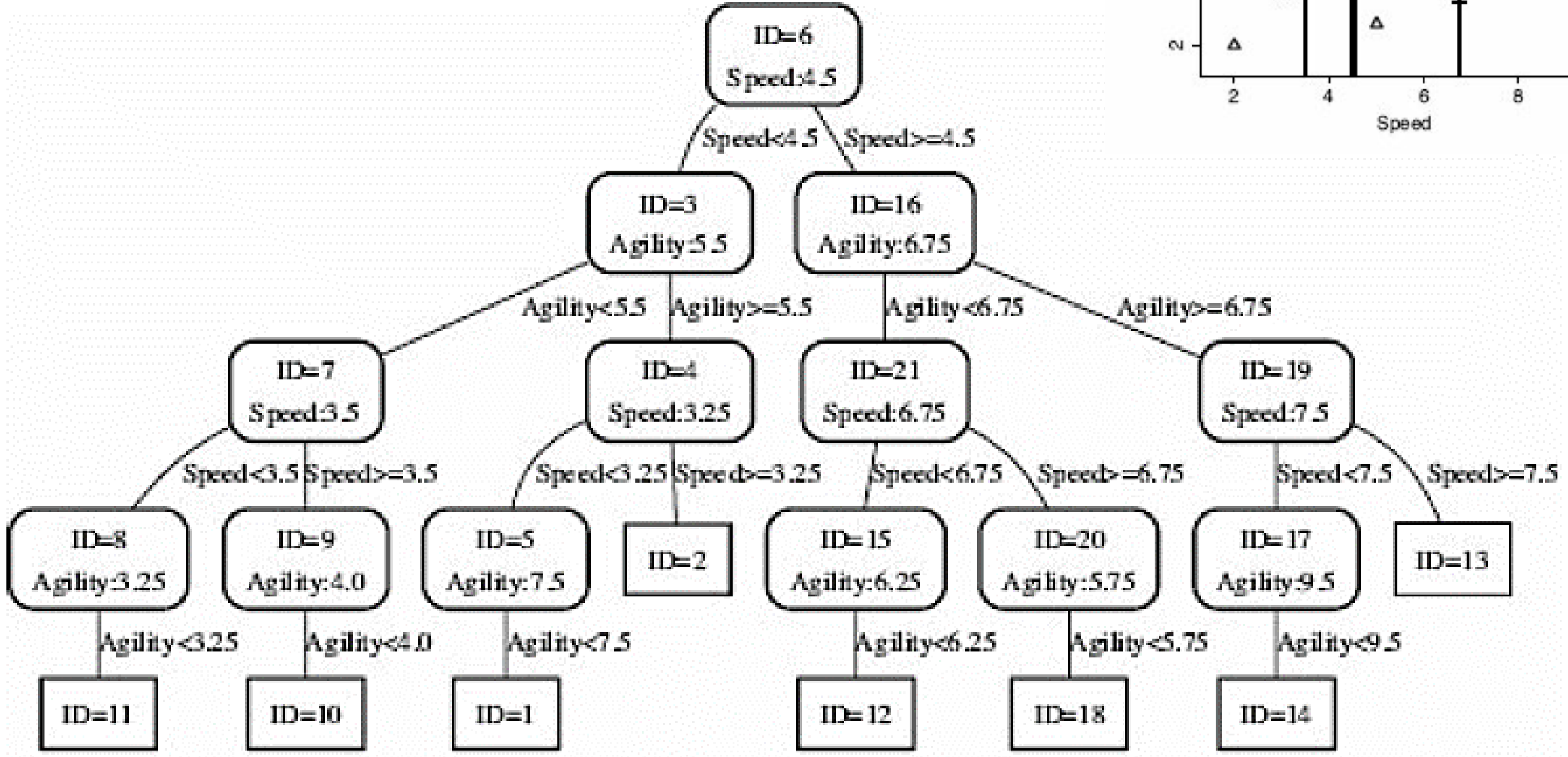
(b)


ID=6
SPEED:4.5
Speed<4.5
Speed≥4.5
ID=3
AGILITY:5.5
IDs= 12,13,14,15,16, 17,18,19,20,21
Agility<5.5
Agility≥5.5
IDs=7,8,9,10,11
IDs=1,2,4,5


Agility
Speed
2
4
6
8

K-d Tree


Agility
Speed
2
4
6
8


ID=6
Speed:4.5
Speed<4.5
Speed>=4.5
ID=3
Agility:5.5
ID=16
Agility:6.75
Agility<5.5
Agility>=5.5
Agility<6.75
Agility>=6.75
ID=7
Speed:3.5
ID=4
Speed:3.25
ID=21
Speed:6.75
ID=19
Speed:7.5
Speed<3.5
Speed>=3.5
Speed<3.25
Speed>=3.25
Speed<6.75
Speed>=6.75
Speed<7.5
Speed>=7.5
ID=8
Agility:3.25
ID=9
Agility:4.0
ID=5
Agility:7.5
ID=2
ID=15
Agility:6.25
ID=20
Agility:5.75
ID=17
Agility:9.5
ID=13
Agility<3.25
Agility<4.0
Agility<7.5
Agility<6.25
Agility<5.75
Agility<9.5
ID=11
ID=10
ID=1
ID=12
ID=18
ID=14

K-d Tree

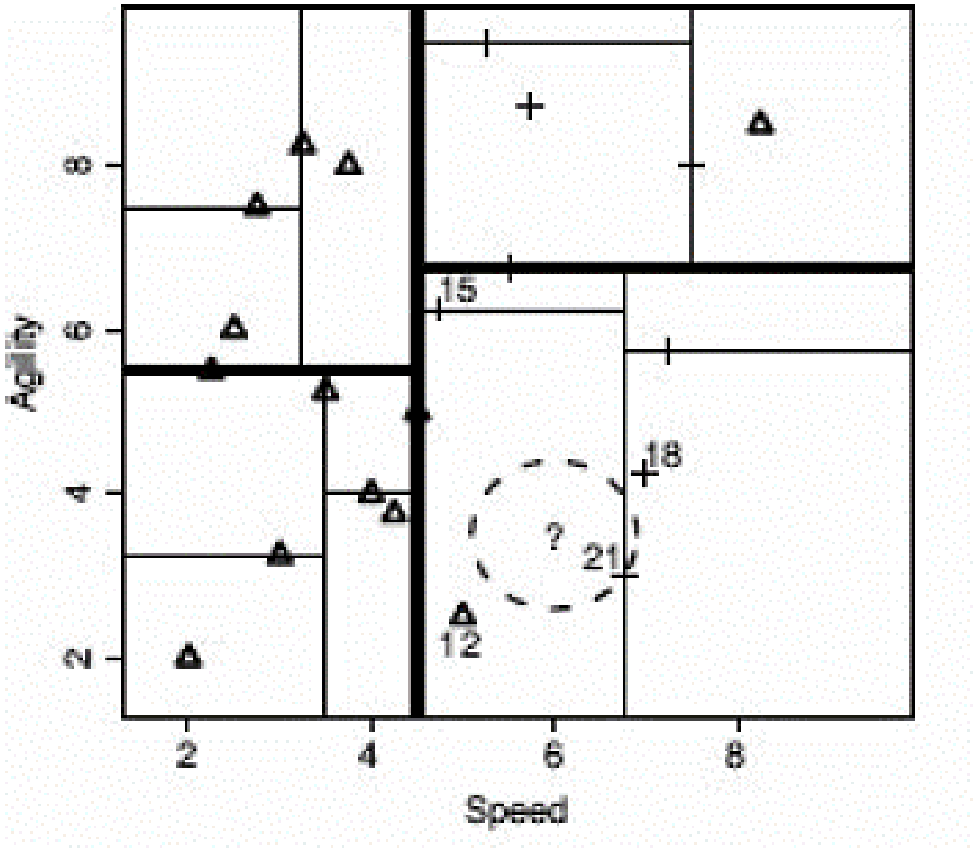
- How to find the nearest neighbor:
  1) Go down the tree based on the values of the instance
  2) Once arrived at a leaf calculate the distance between the instance and the indexed instance of that node (using all features) – *this is our best distance for now*
  3) Go up to the parent node and calculate distance between instance and indexed instance of that node – *if smaller revise best distance*

K-d Tree

- How to find the nearest neighbor:
  4) Check distance between instance and hyperplane represented by the current node (since we did a split based on only one feature, we only use that feature to calculate the distance) – *if lower go down the other branch, if higher go up to parent node*
  5) Do this until we end up at the root again (if from the root we would go up given the rule of the previous step, we are done)
  6) The nearest neighbor is the one with the best distance

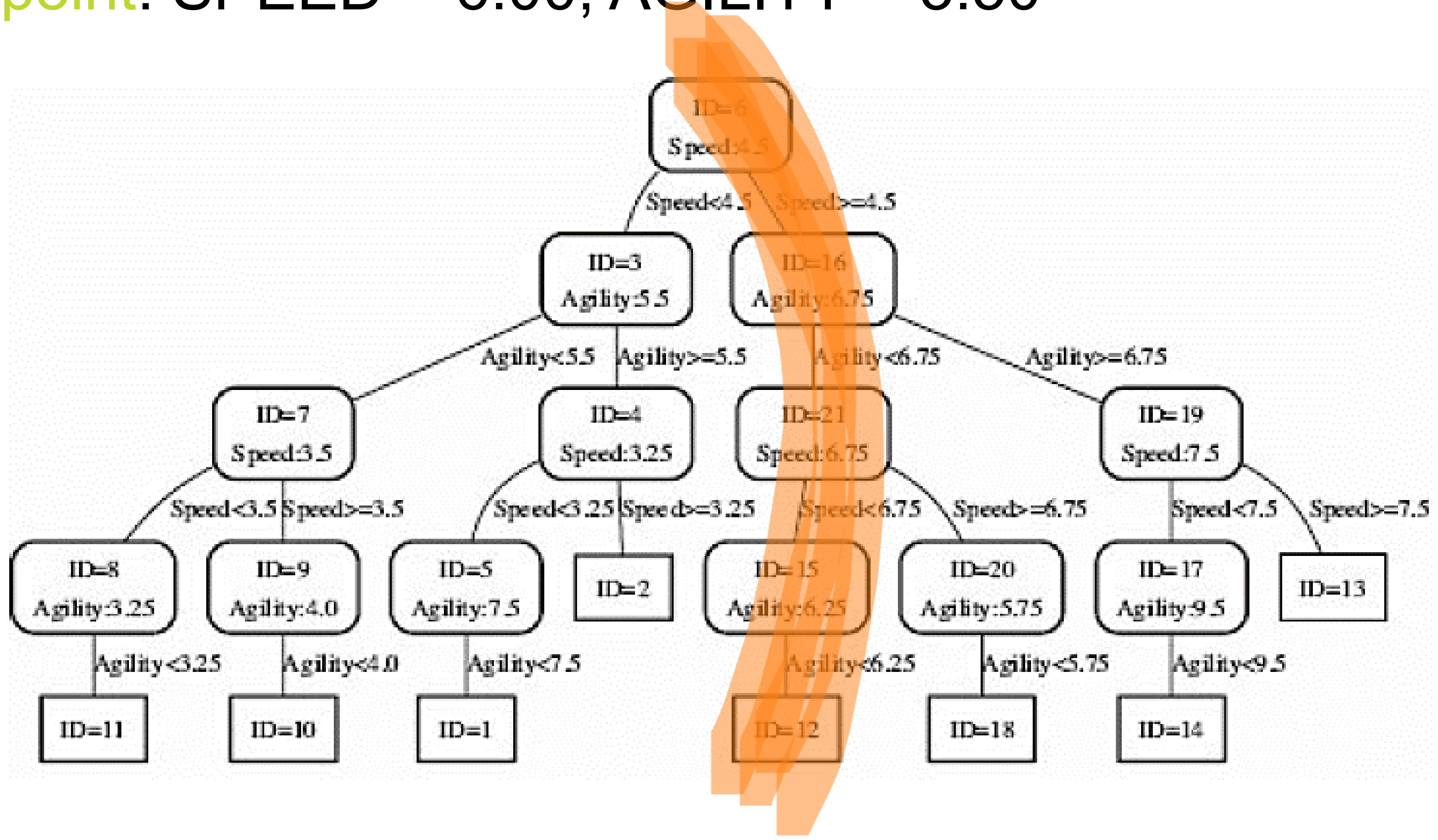
K-d Tree

New datapoint: SPEED = 6.00, AGILITY = 3.50



K-d Tree

New datapoint: SPEED = 6.00, AGILITY = 3.50



node indexes instance d12 (SPEED = 5.00, AGILITY = 2.50)

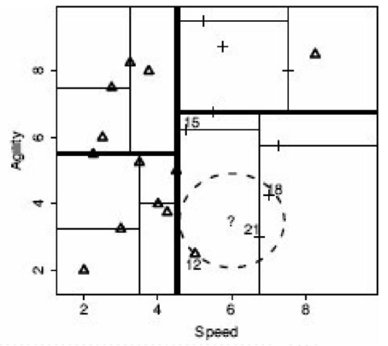
K-d Tree

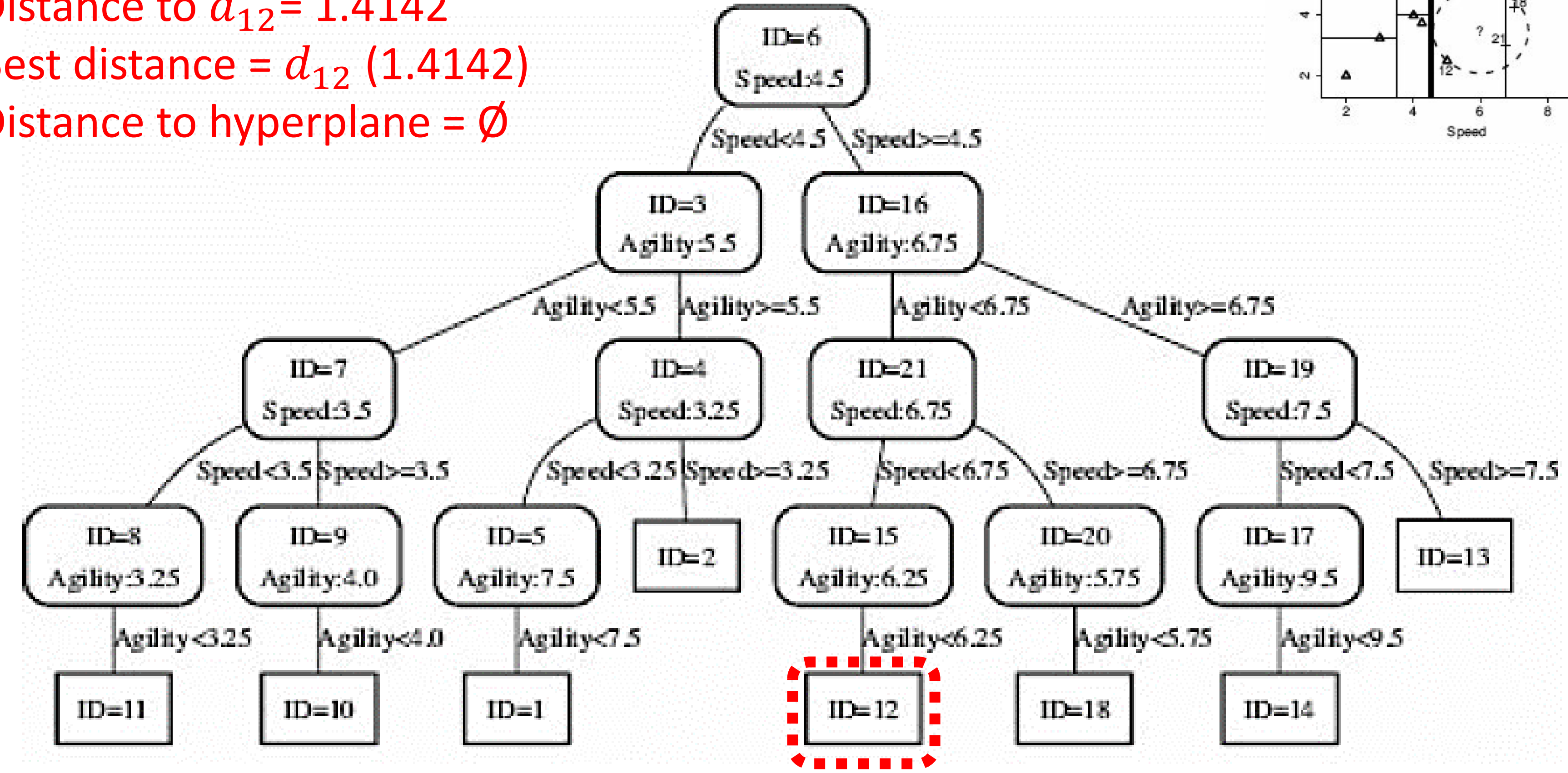
New datapoint: SPEED = 6.00, AGILITY = 3.50

Distance to d_{12}= 1.4142

Best distance = d_{12} (1.4142)

Distance to hyperplane = Ø

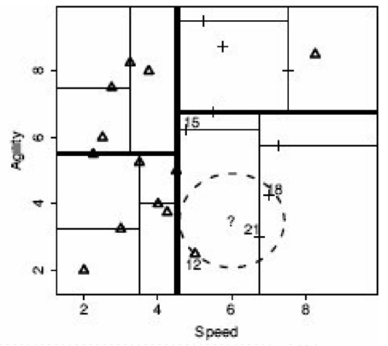


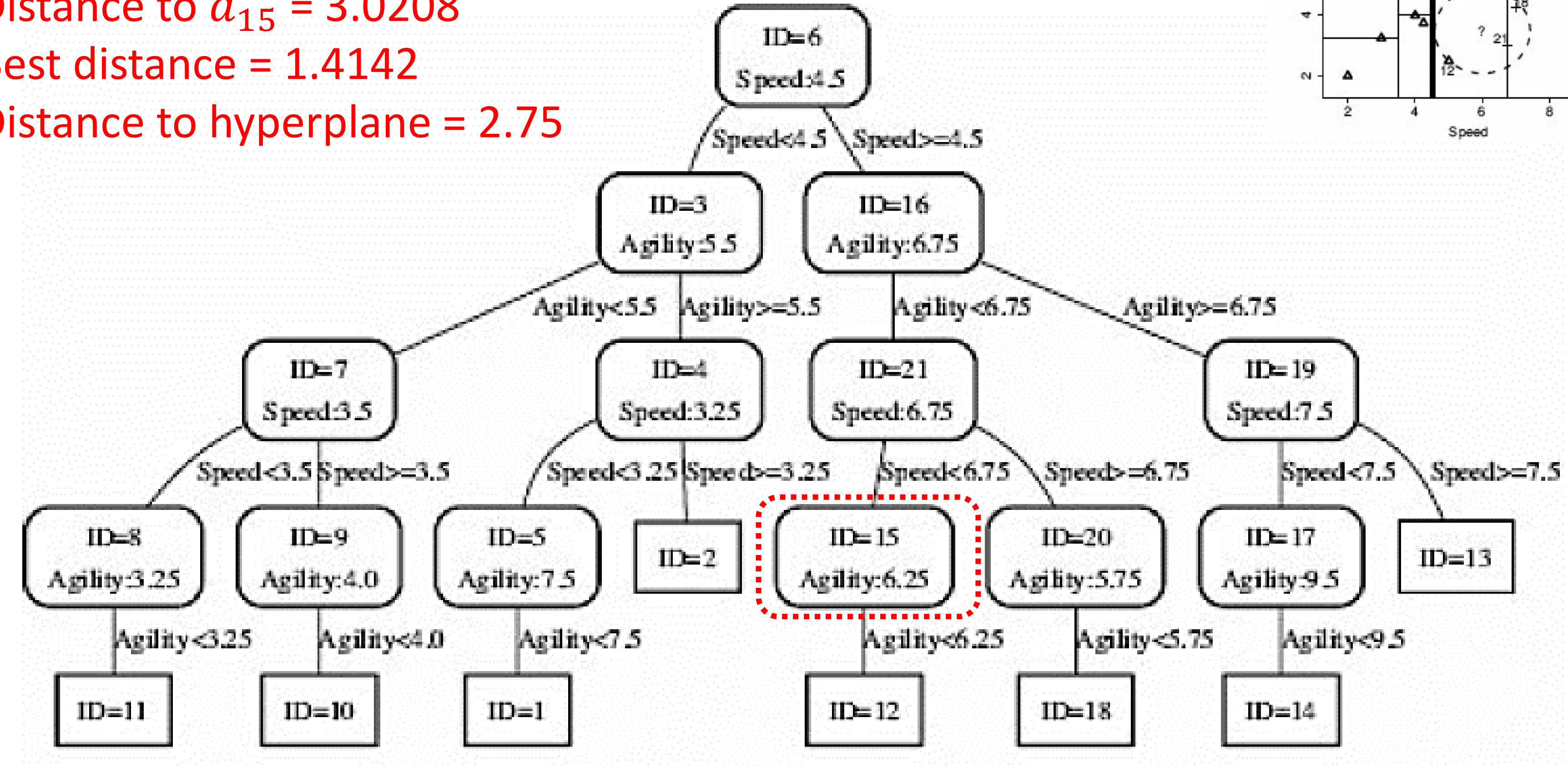


K-d Tree

New datapoint: SPEED = 6.00, AGILITY = 3.50

Distance to d_{15} = 3.0208
Best distance = 1.4142
Distance to hyperplane = 2.75

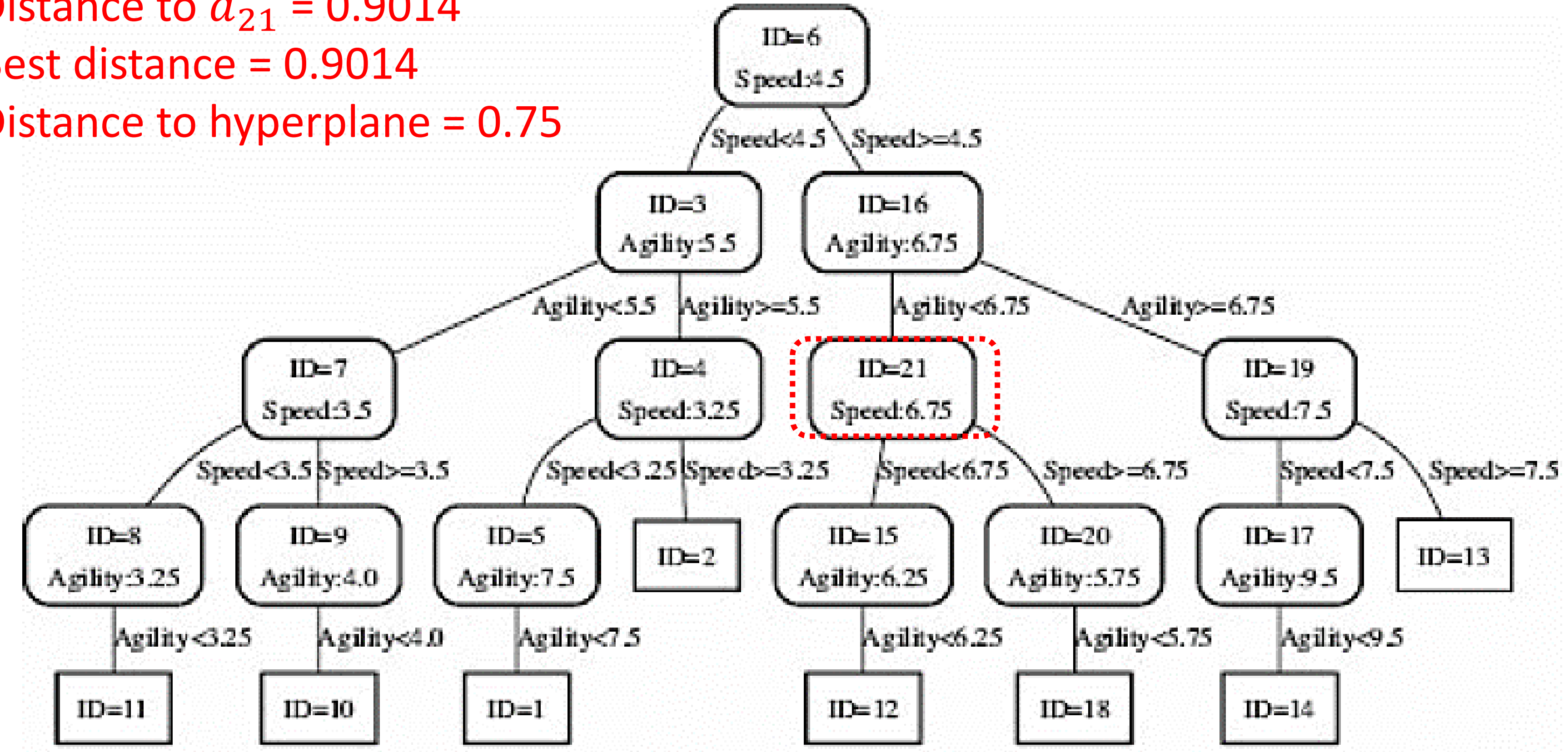




K-d Tree

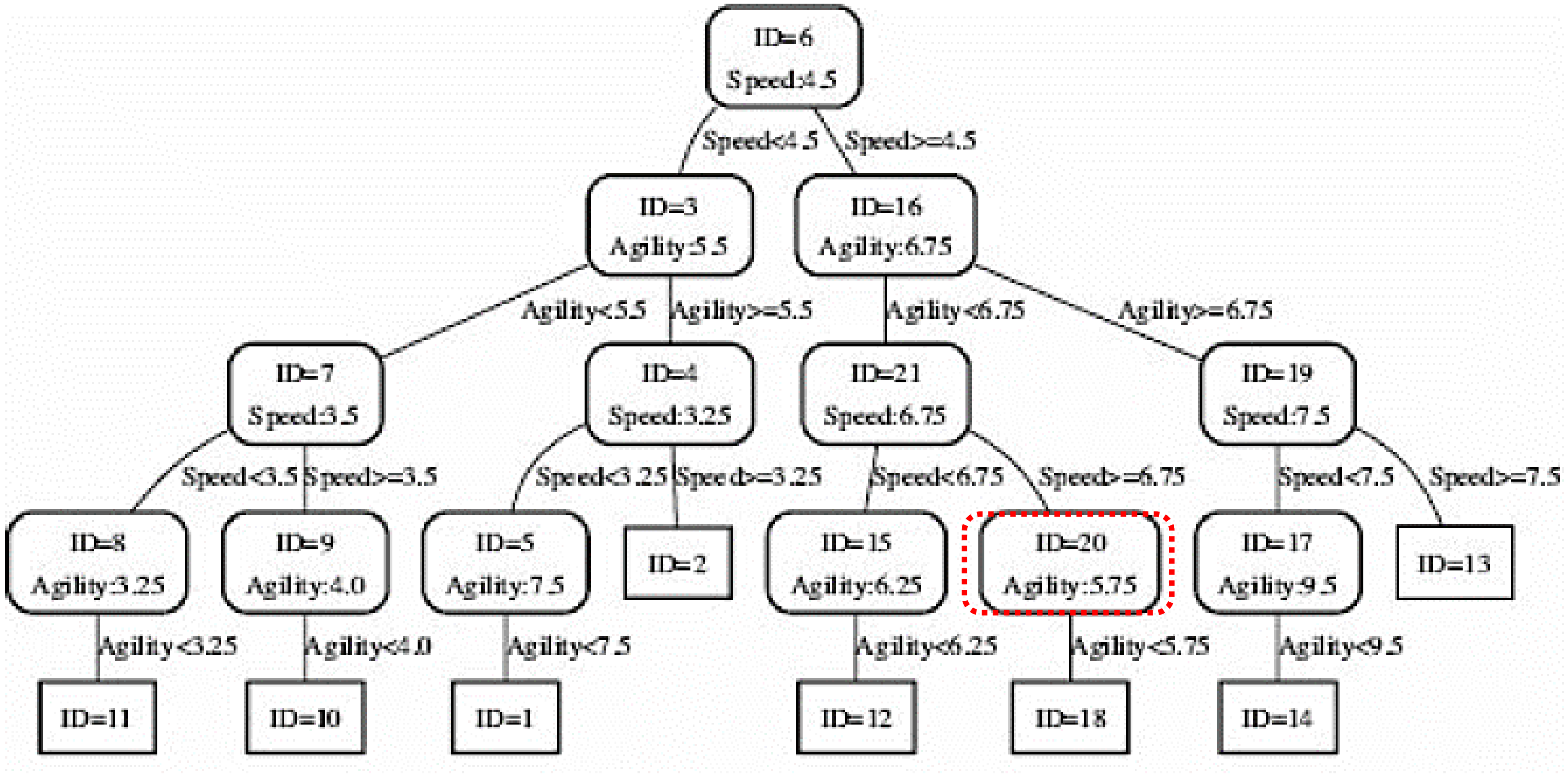
New datapoint: SPEED = 6.00, AGILITY = 3.50

Distance to d_{21} = 0.9014
Best distance = 0.9014
Distance to hyperplane = 0.75



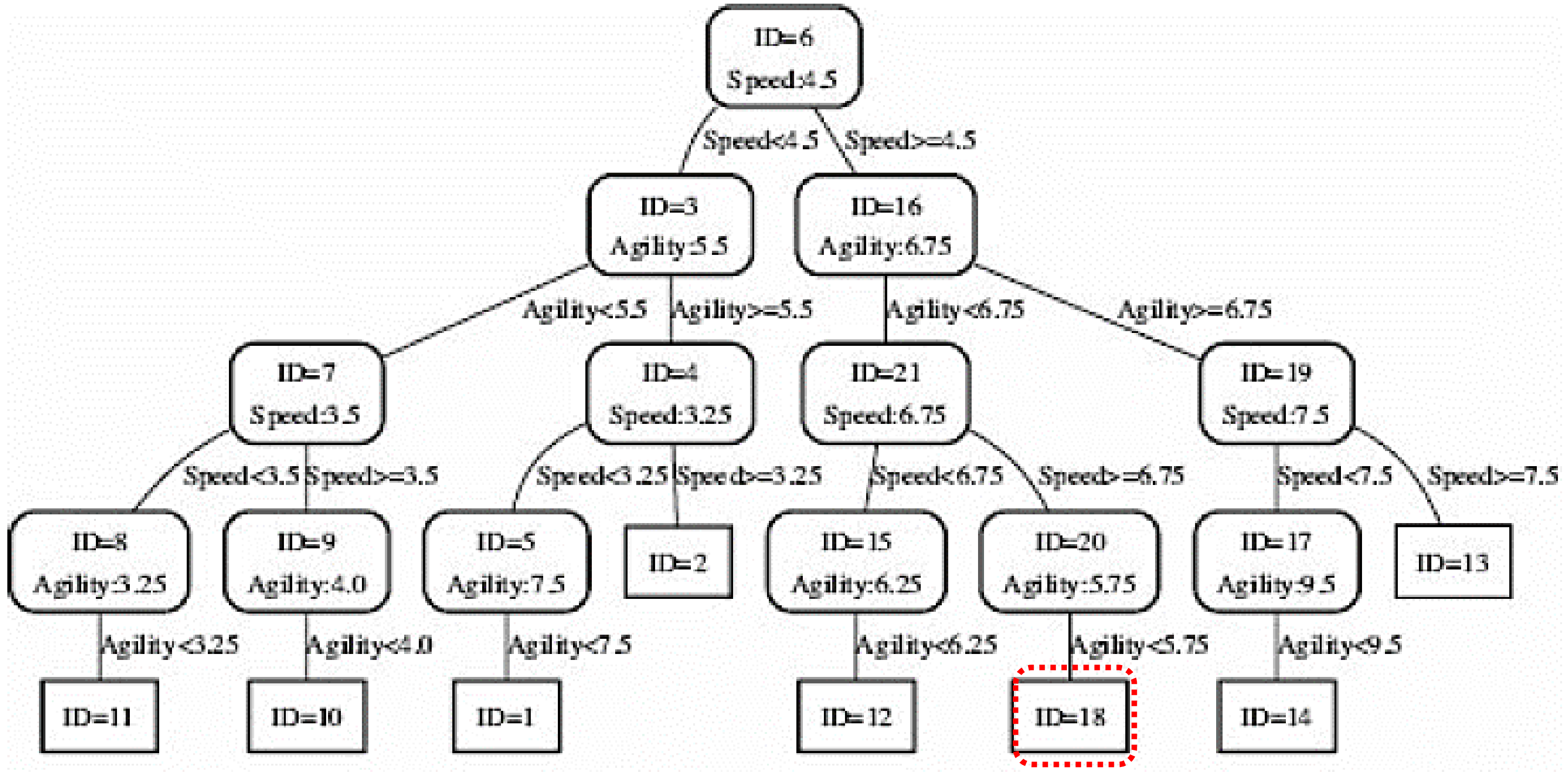
K-d Tree

New datapoint: SPEED = 6.00, AGILITY = 3.50



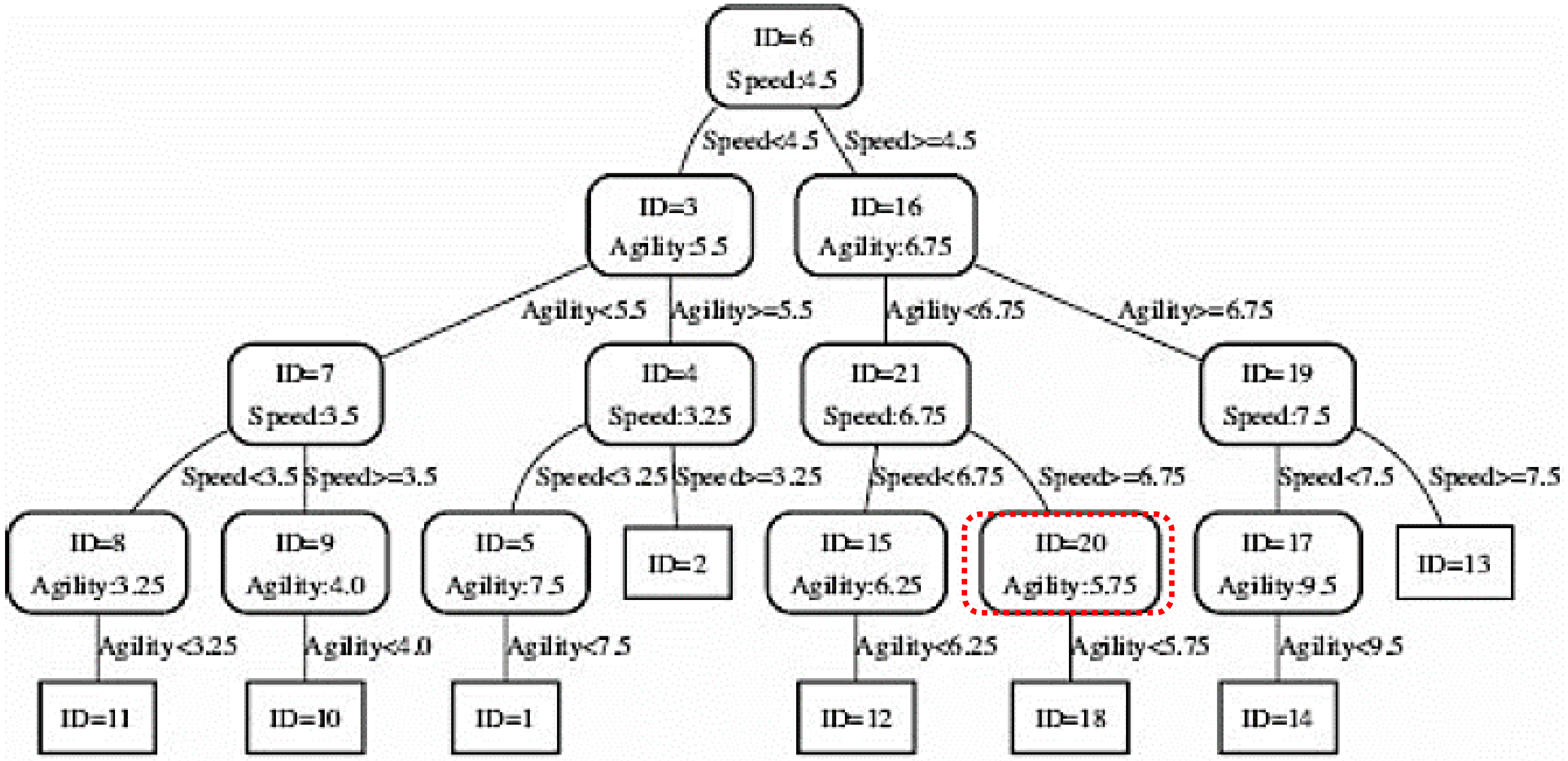
K-d Tree

New datapoint: SPEED = 6.00, AGILITY = 3.50



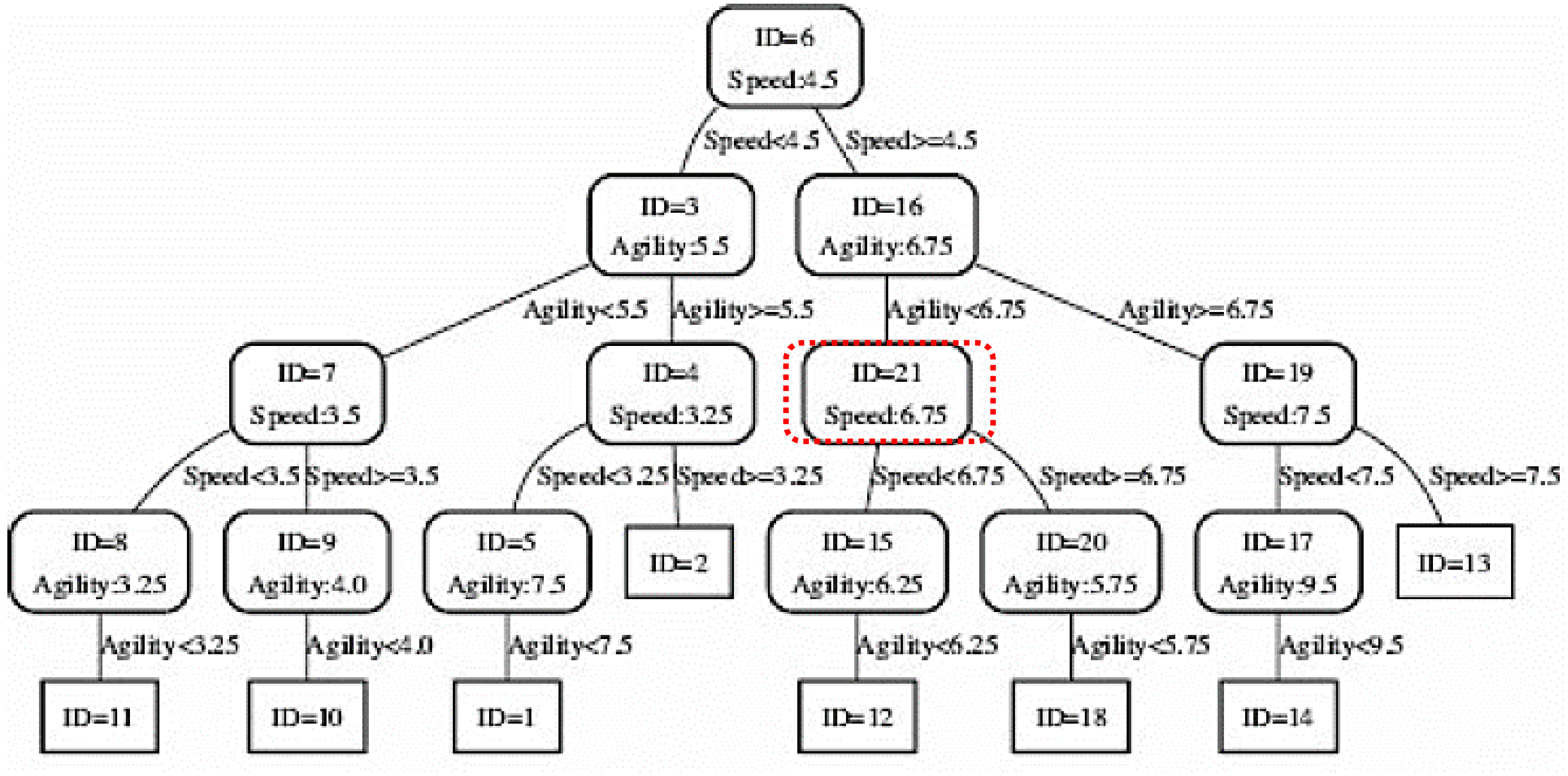
K-d Tree

New datapoint: SPEED = 6.00, AGILITY = 3.50



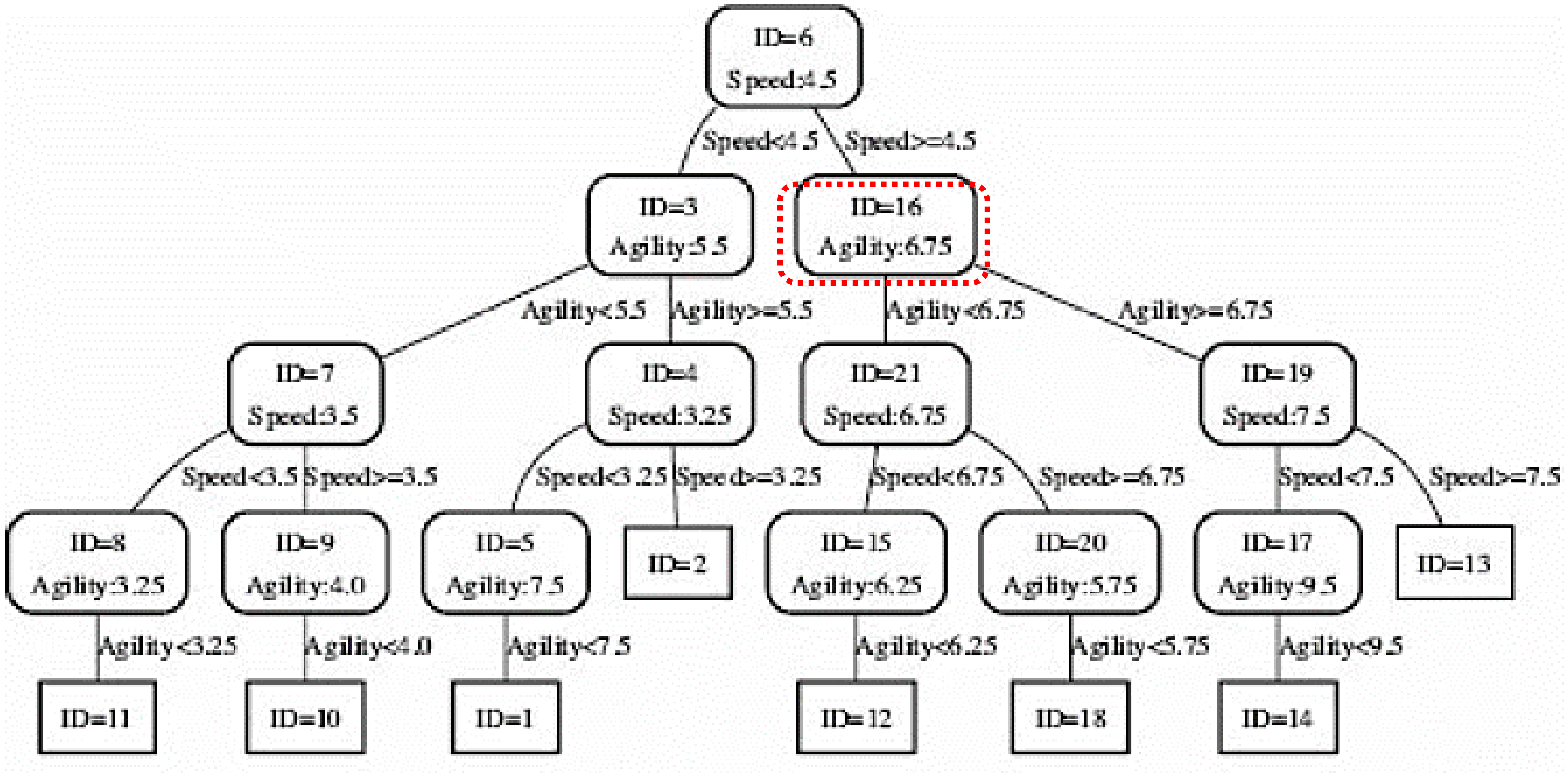
K-d Tree

New datapoint: SPEED = 6.00, AGILITY = 3.50



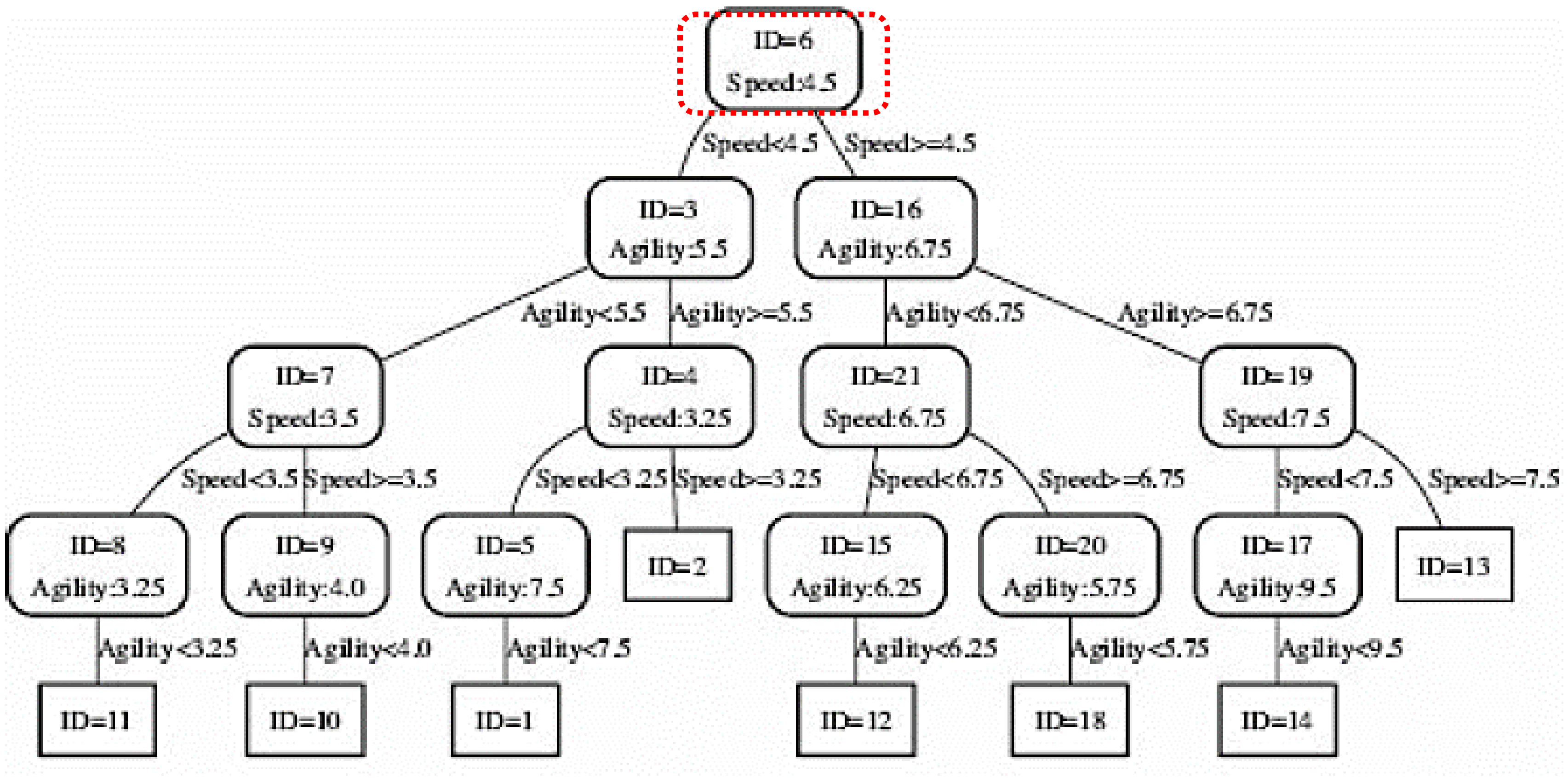
K-d Tree

New datapoint: SPEED = 6.00, AGILITY = 3.50



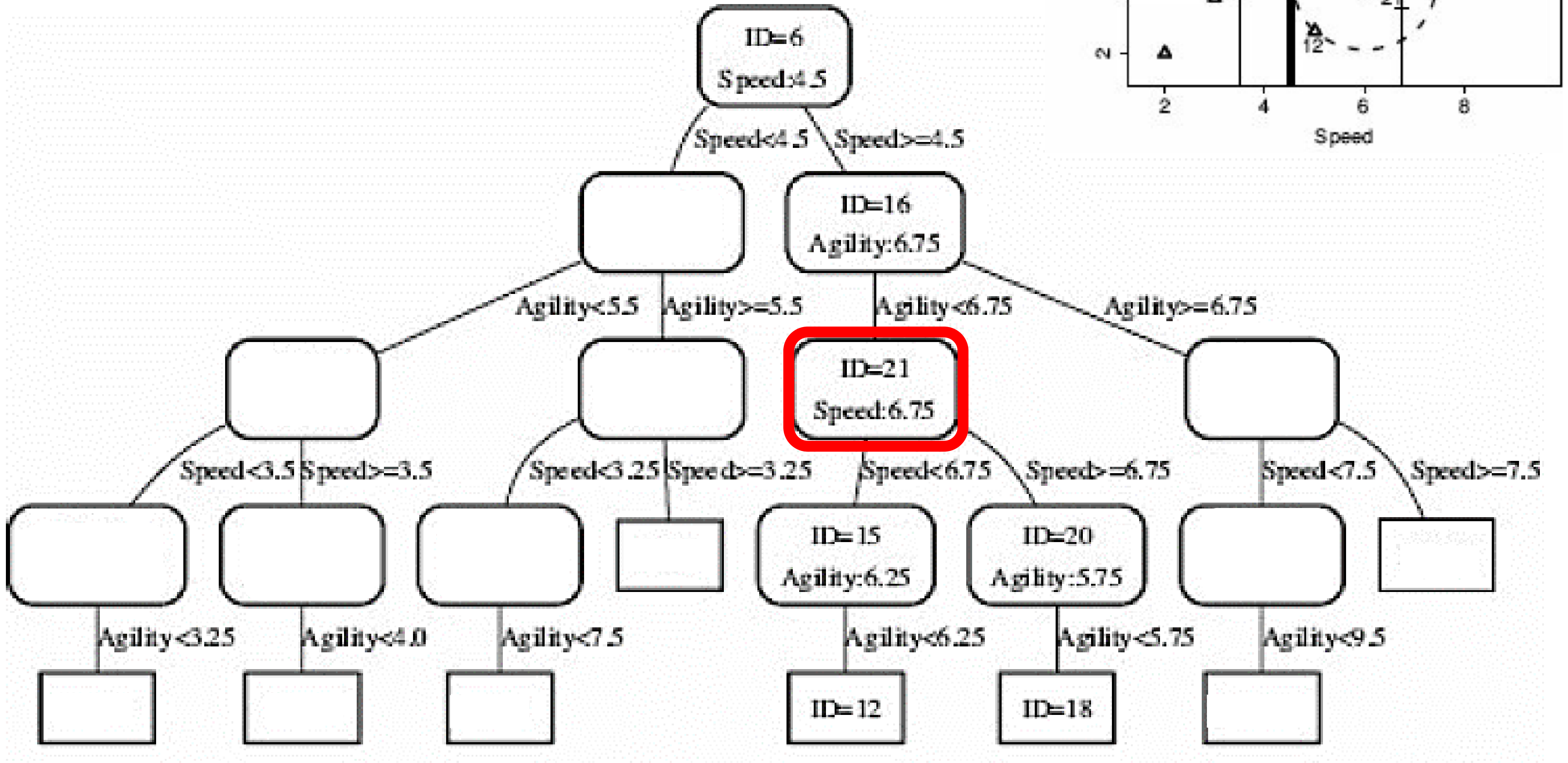
K-d Tree

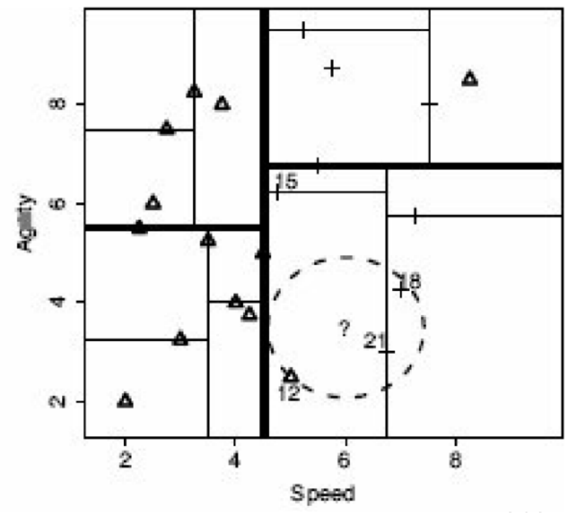
New datapoint: SPEED = 6.00, AGILITY = 3.50



K-d Tree

New datapoint: yes





K-d Tree

K-d trees are very useful when there are many more instances than features!

Thumb rule: 2^m instances for m descriptive features

Continuous Targets

- We can adapt knn to predict continuous target by using the average of the nearest neighbors

Let $x_1, \ldots, x_k$, belong to the list of training_examples,

$$\hat{c}(x_q) \leftarrow \frac{1}{k}\sum_{i=1}^{k} c(x_i)$$

Other Measures of Similarity

Problem with Euclidean measure: can produce counter-intuitive results

Binary descriptive features

1 1 1 1 1 1 1 1 1 1 1 0
0 1 1 1 1 1 1 1 1 1 1 1

d = 1.4142

vs

1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1

d = 1.4142

Other Measures of Similarity

Binary descriptive features

❶ Compare the instance with each training instance:

Co-presence (CP): how often the instance and training instance are both True

Co-absence (CA): how often the instance and training instance are both False

Presence-absence (PA): how often the instance is True but the training instance is False

Absence-presence (AP): how often the instance is False but the training instance is True

Other Measures of Similarity

Binary descriptive features

$$Sim_{RR} = \frac{CP}{total\ number\ of\ binary\ features}$$

❶ Russel-Rao Similarity Index – *uses only CP*

Other Measures of Similarity

Binary descriptive features

$$Sim_{SM} = \frac{CP + CA}{total\ number\ of\ binary\ features}$$

ℹ Sokal-Michener Similarity Index – *uses CP and CA*

Other Measures of Similarity

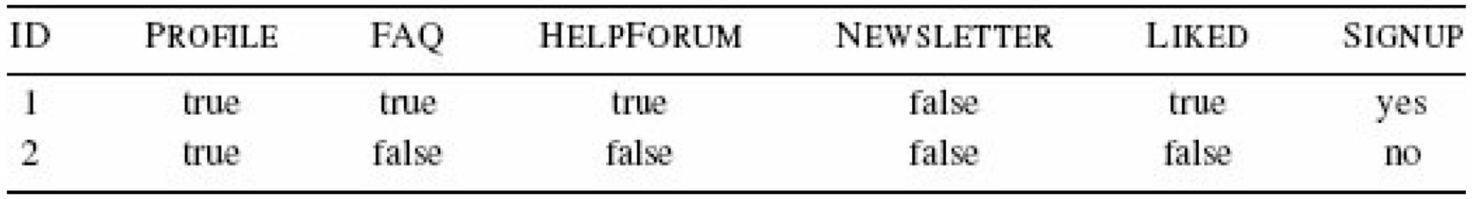
Binary descriptive features

$$Sim_J = \frac{CP}{CP + PA + AP}$$

❶ Jaccard Similarity Index – *uses all but CA*

Other Measures of Similarity

Binary descriptive features



ID	PROFILE	FAQ	HELPFORUM	NEWSLETTER	LIKED	SIGNUP
1	true	true	true	false	true	yes
2	true	false	false	false	false	no

New instance:

PROFILE = *true*, FAQ = *false*, HELPFORUM = *true*,
NEWSLETTER = *false*, LIKED = *false*

Other Measures of Similarity

Binary descriptive features

$$Sim_{RR}(q, d1) = \frac{2}{5} = 0.4$$

$$Sim_{RR}(q, d2) = \frac{1}{5} = 0.2$$

Other Measures of Similarity

Binary descriptive features

$$Sim_{SM}(q, d1) = \frac{3}{5} = 0.6$$

$$Sim_{SM}(q, d2) = \frac{4}{5} = 0.8$$

Other Measures of Similarity

Binary descriptive features

$$Sim_J(q, d1) = \frac{2}{4} = 0.5$$

$$Sim_J(q, d2) = \frac{1}{2} = 0.5$$

Other Measures of Similarity

Continuous descriptive features

ℹ Sometimes we want to compare instances based on their relative behaviour and the absolute values don’t matter that much

Cosine Similarity

Other Measures of Similarity

Continuous descriptive features

Cosine Similarity: we project the data onto a circle (hypersphere) with radius equal to one and compare the cosine.

ℹ The cosine similarity ranges from 0 to 1 (if we normalize the data to a positive range) and the closer to 1, the more similar are the instances!

Other Measures of Similarity

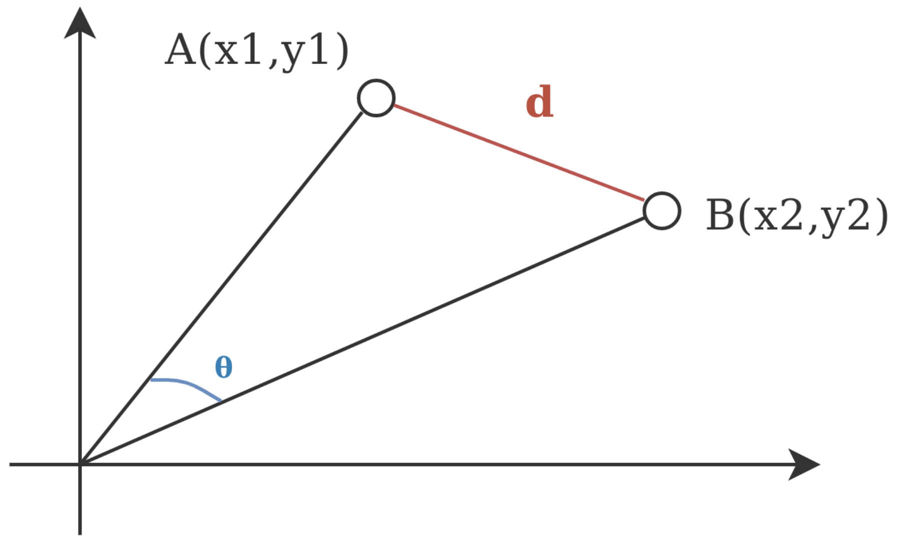
Continuous descriptive features

$$Sim_{cosine}(a, b) = \frac{a \cdot b}{\sqrt{a^2} \times \sqrt{b^2}}$$

Other Measures of Similarity

Continuous descriptive features

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$



Other Measures of Similarity

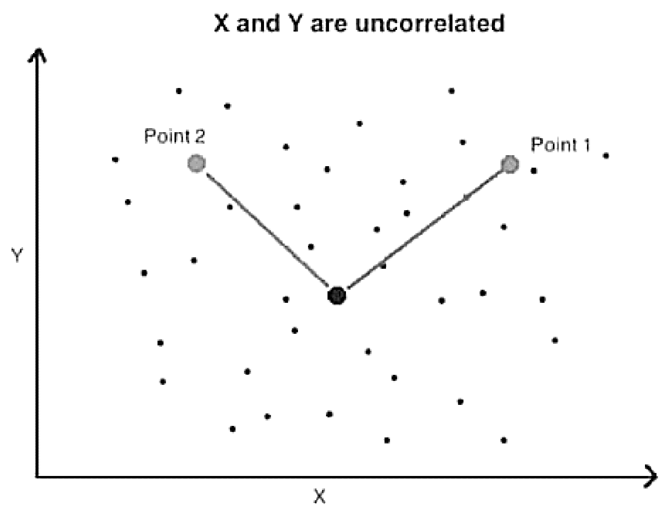
Continuous descriptive features

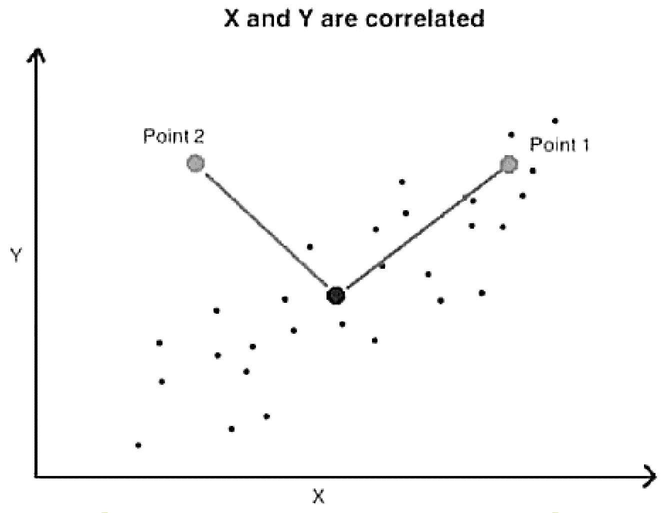
❶ There is a distance measure that uses covariance to take into account how spread out the instances are when judging similarity

Mahalanobis Distance

Other Measures of Similarity

Continuous descriptive features





same distance

because of the way the data is expected to behave, de distance to Point 1 is lower than the distance to Point 2

NOVA IMS INFORMATION MANAGEMENT SCHOOL