# Instance based classification

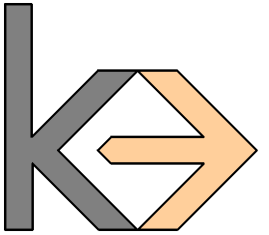
TECHNISCHE UNIVERSITÄT DARMSTADT

# Data Mining and Machine Learning: Techniques and Algorithms

**Eneldo Loza Mencía**

*eneldo@ke.tu-darmstadt.de*

Knowledge Engineering Group, TU Darmstadt



International Week 2019, 21.1. – 24.1.
University of Economics, Prague

# Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

- Rote Learning
- k-Nearest Neighbor Classification
  - Choosing k
  - Distance functions
  - Instance and Feature Weighting
  - Efficiency
- kD-Trees
  - Ball trees

# Rote Learning

TECHNISCHE UNIVERSITÄT DARMSTADT

| Day | Temperature | Outlook | Humidity | Windy | Play Golf? |
|---|---|---|---|---|---|
| 07-05 | hot | sunny | high | false | no |
| 07-06 | hot | sunny | high | true | no |
| 07-07 | hot | overcast | high | false | yes |
| 07-09 | cool | rain | normal | false | yes |
| 07-10 | cool | overcast | normal | true | yes |
| 07-12 | mild | sunny | high | false | no |
| 07-14 | cool | sunny | normal | false | yes |
| 07-15 | mild | rain | normal | false | yes |
| 07-20 | mild | sunny | normal | true | yes |
| 07-21 | mild | overcast | high | true | yes |
| 07-22 | hot | overcast | normal | false | yes |
| 07-23 | mild | rain | high | true | no |
| 07-26 | cool | rain | normal | true | no |
| 07-30 | mild | rain | high | false | yes |
| today | cool | sunny | normal | false | |

# Instance Based Classifiers

TECHNISCHE UNIVERSITÄT DARMSTADT

- No model is learned
  - The stored training instances themselves represent the knowledge
  - Training instances are searched for instance that most closely resembles new instance

  → *lazy learning*

- Examples:
  - Rote-learner
    - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
  - Nearest-neighbor classifier
    - Uses k "closest" points (nearest neigbors) for performing classification

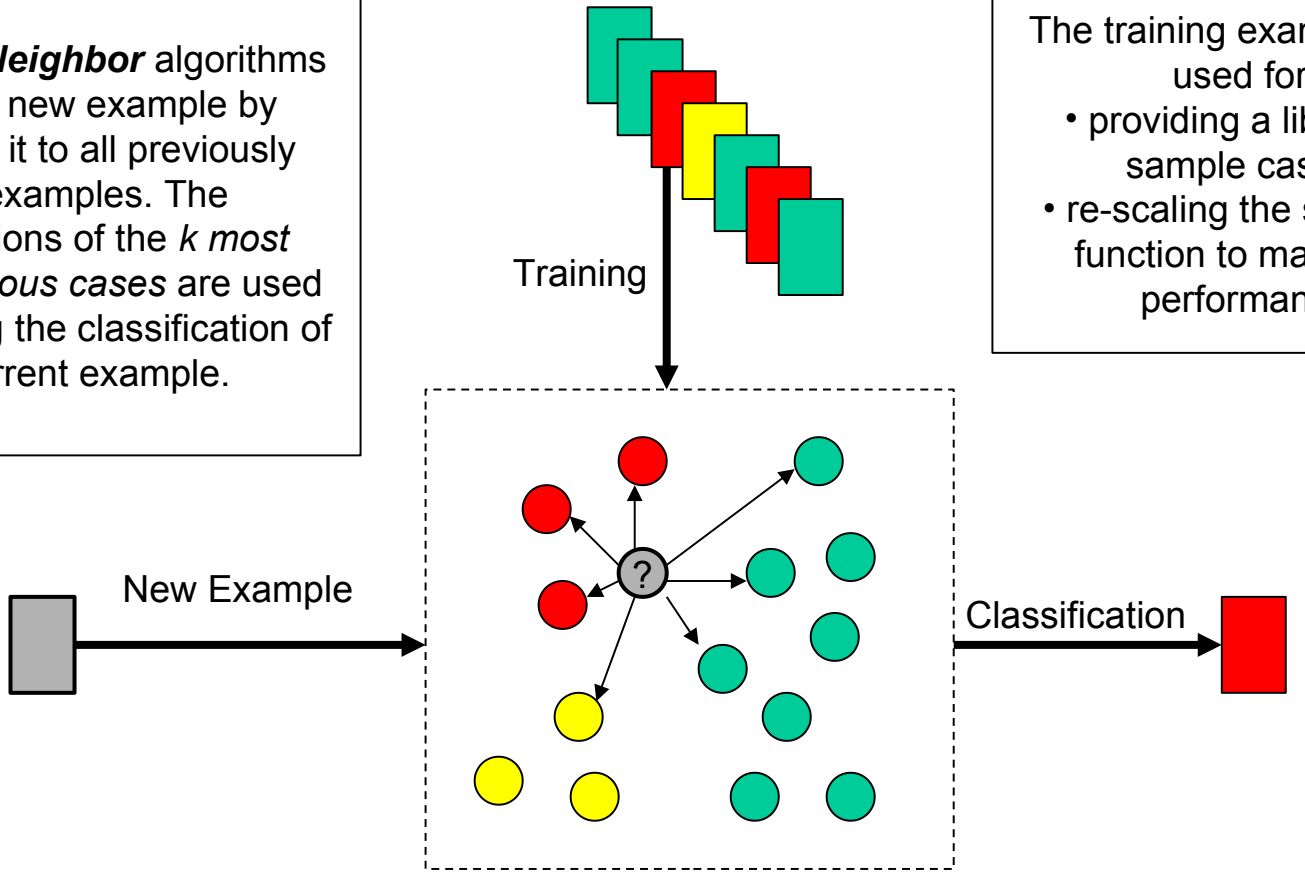
# Nearest Neighbor Classification

TECHNISCHE UNIVERSITÄT DARMSTADT

| Day | Temperature | Outlook | Humidity | Windy | Play Golf? |
|---|---|---|---|---|---|
| 07-05 | hot | sunny | high | false | no |
| 07-06 | hot | sunny | high | true | no |
| 07-07 | hot | overcast | high | false | yes |
| 07-09 | cool | rain | normal | false | yes |
| 07-10 | cool | overcast | normal | true | yes |
| 07-12 | mild | sunny | high | false | no |
| 07-14 | cool | sunny | normal | false | yes |
| 07-15 | mild | rain | normal | false | yes |
| 07-20 | mild | sunny | normal | true | yes |
| 07-21 | mild | overcast | high | true | yes |
| 07-22 | hot | overcast | normal | false | yes |
| 07-23 | mild | rain | high | true | no |
| 07-26 | cool | rain | normal | true | no |
| 12-30 | mild | rain | high | false | yes |
| tomorrow | mild | sunny | normal | false | yes |

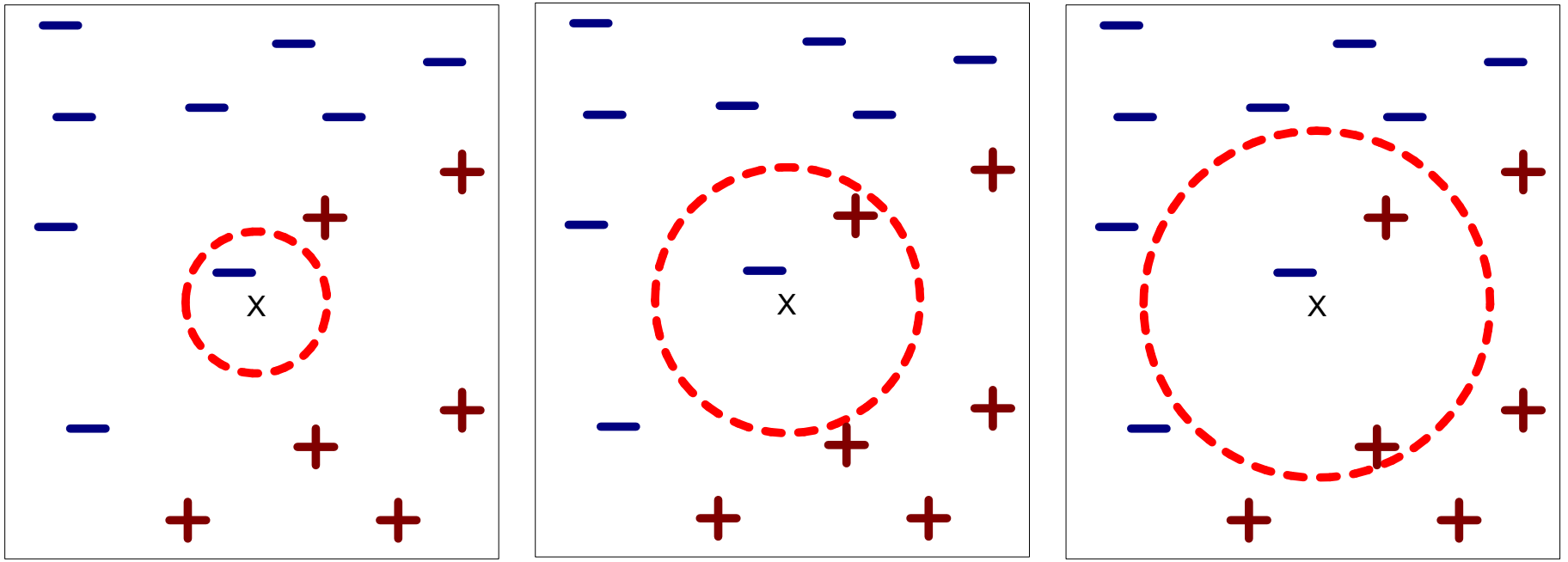
# Nearest Neighbor Classifier

***K-Nearest Neighbor*** algorithms classify a new example by comparing it to all previously seen examples. The classifications of the *k most similar previous cases* are used for predicting the classification of the current example.




The training examples are used for
- providing a library of sample cases
- re-scaling the similarity function to maximize performance

# Nearest Neighbors




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

*k* nearest neighbors of an example $x$ are the data points that have the $k$ smallest distances to $x$

# Prediction

TECHNISCHE UNIVERSITÄT DARMSTADT

The predicted class is determined from the nearest neighbor list

- classification
  - take the majority vote of class labels among the k-nearest neighbors

$$\hat{y} = \max_c \sum_{i=1}^{k} \begin{cases} 1 & \text{if} \quad y_i = c \\ 0 & \text{if} \quad y_i \neq c \end{cases} = \max_c \sum_{i=1}^{k} \mathbf{1}(y_i = c)$$

indicator function

- can be easily be extended to regression
  - predict the average value of the class value of the k-nearest neighbors

$$\hat{y} = \frac{1}{k} \sum_{i=1}^{k} y_i$$

# Weighted Prediction

TECHNISCHE UNIVERSITÄT DARMSTADT

- Often prediction can be improved if the influence of each neighbor is weighted

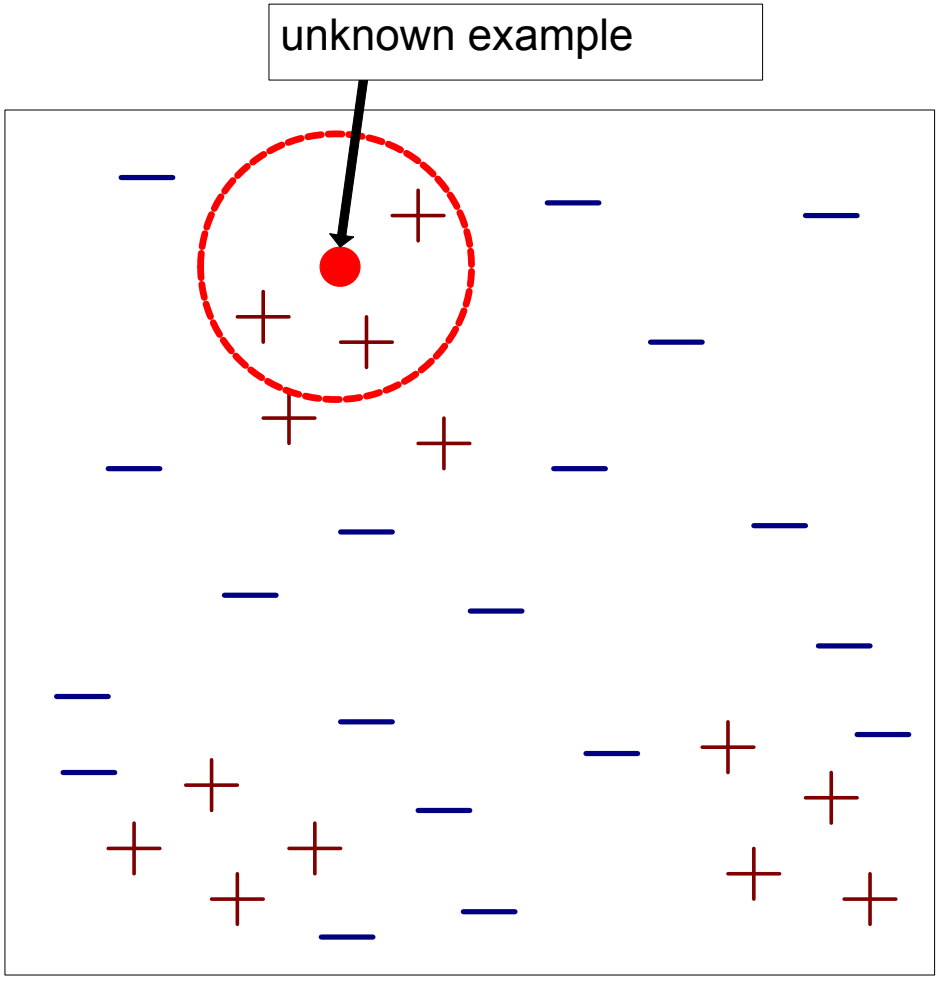
$$\hat{y} = \frac{\sum_{i=1}^{k} w_i \cdot y_i}{\sum_{i=1}^{k} w_i}$$

- Weights typically depend on distance, e.g.

$$w_i = \frac{1}{d(x_i, x)^2}$$

- Note:
  - with weighted distances, we could use all examples for classifications (→ Inverse Distance Weighting)

# Nearest-Neighbor Classifiers



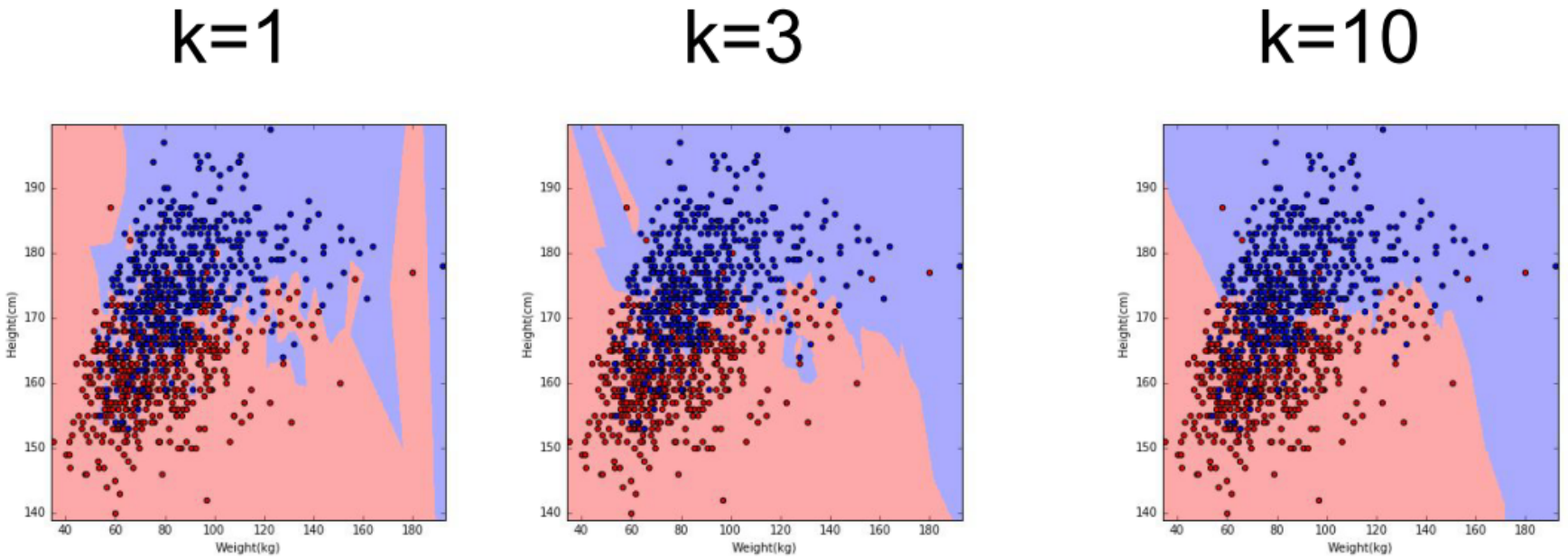

- Require three things
  - The set of stored examples
  - Distance Metric to compute distance between examples
  - The value of *k*, the number of nearest neighbors to retrieve
- To classify an unknown example:
  - Compute distance to other training examples
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown example
    (e.g., by taking majority vote)

# Lazy Learning Algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- kNN is considered a lazy learning algorithm
  - Defers data processing until it receives a request to classify an unlabelled example
  - Replies to a request for information by combining its stored training data
  - Discards the constructed answer and any intermediate results
- Other names for lazy algorithms
  - Memory-based, Instance-based , Exemplar-based , Case-based, Experience based
- This strategy is opposed to eager learning algorithms which
  - Compiles its data into a compressed description or model
  - Discards the training data after compilation of the model
  - Classifies incoming patterns using the induced model

# Choosing the value of k




- if k is too small
  - sensitive to noise in the data (misclassified examples)
- greater k leads to smoother boundaries

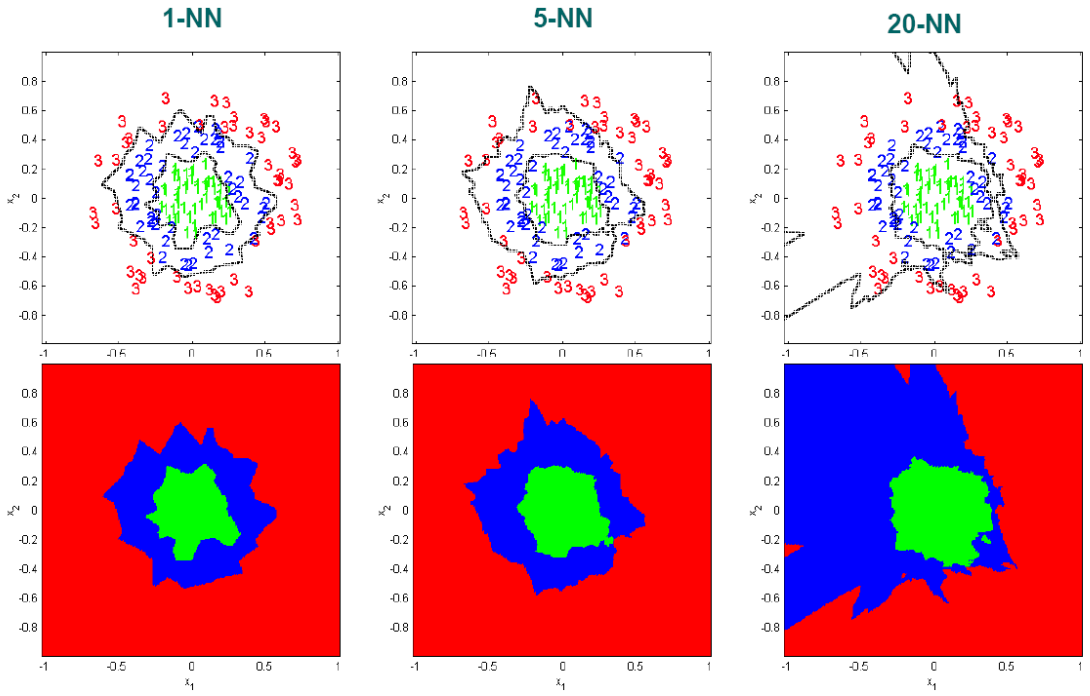


Steven Skiena

# Choosing the value of k

TECHNISCHE UNIVERSITÄT DARMSTADT

- If k is too large
  - neighborhood may include points from other classes
  - limiting case:
    - all examples are considered
    - largest class is predicted




- good values can be found
  - e.g, by evaluating various values with cross-validation on the training data

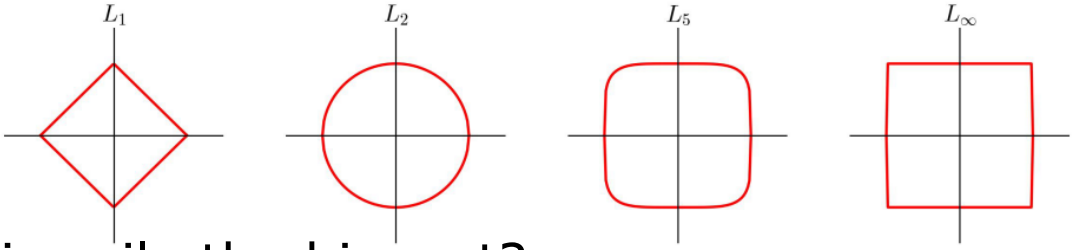
# Distance Functions

TECHNISCHE UNIVERSITÄT DARMSTADT

- Computes the distance between two examples
  - so that we can find the "nearest neighbor" to a given example
- General Idea:
  - reduce the distance $d(x_1, x_2)$ of two examples to the distances $d_A(v_1, v_2)$ between two values for attribute $A$
- Popular choices
  - Euclidean Distance (L2):
    - straight-line between two points
  - Manhattan or City-block Distance (L∞):
    - sum of axis-parallel line segments
- other choices possible
  - important in higher dimensional spaces: do we care about deviations in all dimensions or primarily the biggest?

$$d(x_1, x_2) = \sqrt{\sum_A d_A(v_{1,A}, v_{2,A})^2}$$

$$d(x_1, x_2) = \sum_A d_A(v_{1,A}, v_{2,A})$$



# Distance Functions for Numerical Attributes

TECHNISCHE UNIVERSITÄT DARMSTADT

- Numerical Attributes:
  - distance between two attribute values

$$d_A(v_1, v_2) = |v_1 - v_2|$$

- Normalization:
  - Different attributes are measured on different scales

    → values need to be normalized in [0,1]:

$$\hat{v}_i = \frac{v_i - \min v_j}{\max v_j - \min v_j}$$

  - Note:
    - This normalization assumes a (roughly) uniform distribution of attribute values
    - For other distributions, other normalizations might be preferable
      - e.g.: logarithmic for salaries?

# Distance Functions for Symbolic Attributes

TECHNISCHE UNIVERSITÄT DARMSTADT

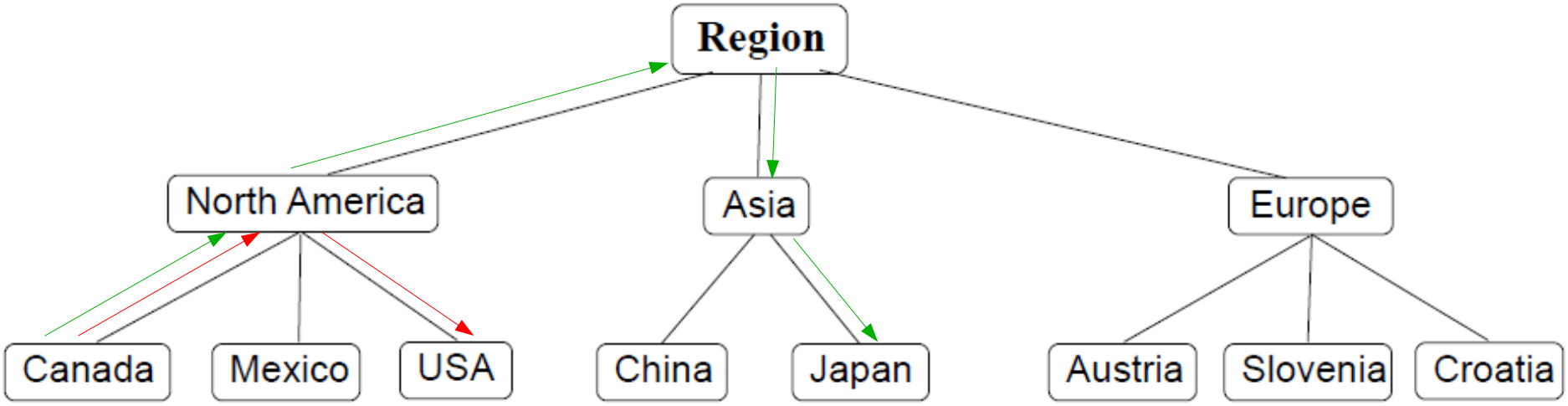
- 0/1 distance

$$d_A(v_1, v_2) = \begin{cases} 0 & \text{if} \quad v_1 = v_2 \\ 1 & \text{if} \quad v_1 \neq v_2 \end{cases}$$

- Value Difference Metric (VDM) (Stanfill & Waltz 1986)
  - two values are similar if they have approximately the same distribution over all classes (similar relative frequencies in all classes)

# Other Distance Functions

- Other distances are possible
  - hierarchical attributes
    - distance of the values in the hierarchy
    - e.g., length of shortest path form $v_1$ to $v_2$




$$d(\text{Canada}, \text{USA}) = 2, \; d(\text{Canada}, \text{Japan}) = 4$$

# Other Distance Functions

TECHNISCHE UNIVERSITÄT DARMSTADT

- Other distances are possible
  - hierarchical attributes
    - distance of the values in the hierarchy
    - e.g., length of shortest path form $v_1$ to $v_2$
- in general
  - distances are domain-dependent
  - can be chosen appropriately

## Distances for Missing Values

- not all attribute values may be specified for an example
- Common policy:
  - assume missing values to be maximally distant

# Feature and Instance Weighting

TECHNISCHE UNIVERSITÄT DARMSTADT

- Feature Weighting
  - Not all dimensions are equally important
    - comparisons on some dimensions might even be completely irrelevant for the prediction task
    - straight-forward distance functions give equal weight to all dimensions
    - RELIEF: give higher weight to features which allow to better discriminate between classes (Kira & Rendell, ICML-92)
- Instance Weighting:
  - we assign a weight to each instance
  - instances with lower weights are always distant
  - hence have a low impact on classification
  - PEBLS (Cost & Salzberg, 1993):

$$d'(x_1, x_2) = \frac{1}{w_{x_1} \cdot w_{x_2}} \cdot d(x_1, x_2) \qquad w_x = \frac{\text{Number of times } x \text{ has correctly predicted the class}}{\text{Number of times } x \text{ has been used for prediction}}$$

# Efficiency of NN algorithms

TECHNISCHE UNIVERSITÄT DARMSTADT

- very efficient in training
  - only store the training data
- not so efficient in testing
  - computation of distance measure to every training example
  - much more expensive than, e.g., decision trees

- Note that k-NN and 1-NN are equal in terms of efficiency
  - retrieving the k nearest neighbors is (almost) not more expensive than retrieving a single nearest neighbor
  - k nearest neighbors can be maintained in a queue

# Finding nearest neighbors efficiently

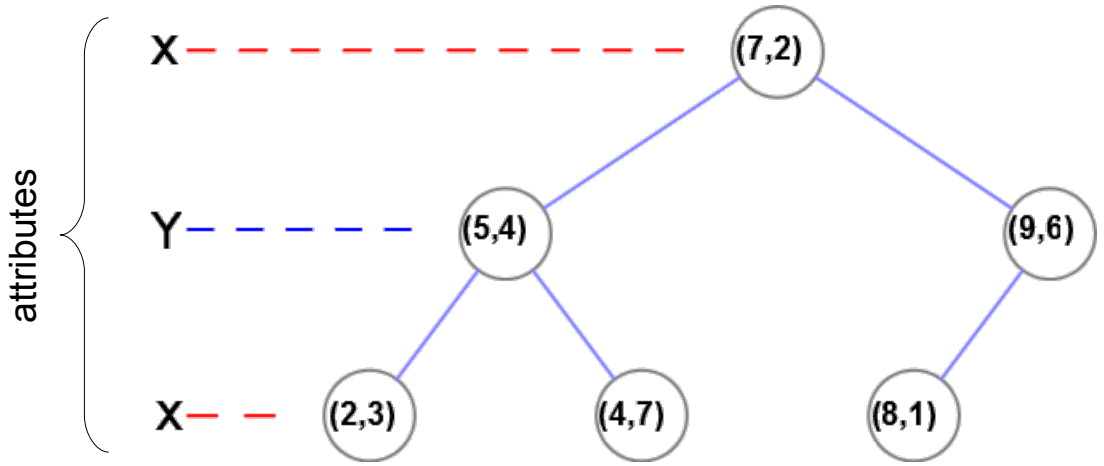
TECHNISCHE UNIVERSITÄT DARMSTADT

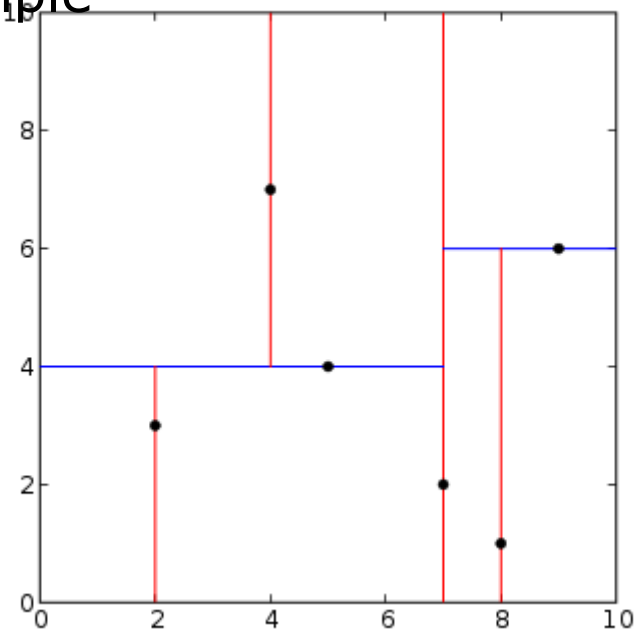
- Simplest way of finding nearest neighbour:
  - linear scan of the data
  - classification takes time proportional to the product of the number of instances in training and test sets

- Nearest-neighbor search can be done more efficiently using appropriate data structures
  - kD-trees
  - ball trees

# kD-Trees

- common setting (others possible)
  - each level corresponds to one of the attributes
    - order of attributes can be arbitrary, fixed, and cyclic
  - each level splits according to its attribute
    - ideally use the median value (results in balanced trees)
    - often simply use the value of the next example



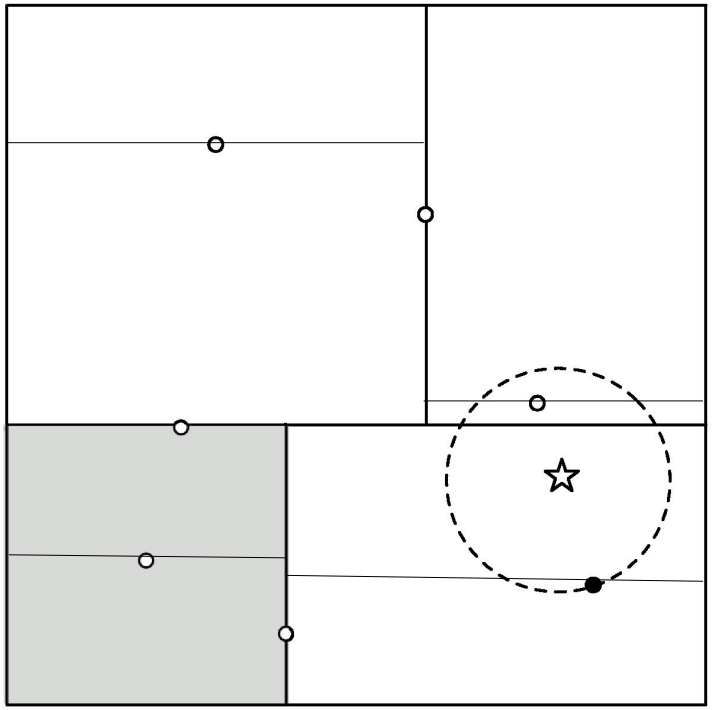



# Using kD-trees: example

TECHNISCHE UNIVERSITÄT DARMSTADT

- The effect of a kD-tree is to partition the (multi-dimensional) sample space according to the underlying data distribution
  - finer partitioning in regions with high density
  - coarser partitioning in regions with low density
- For a given query point
  - descending the tree to find the data points lying in the cell that contains the query point
  - examine surrounding cells if they overlap the ball centered at the query point and the closest data point so far
    - recursively back up one level and check distance to the split point
    - if overlap also search other branch

→ only a few cells have to be searched



# Using kD-trees: example

- Assume we have example [1,5]
  - Unweighted Euclidian distance

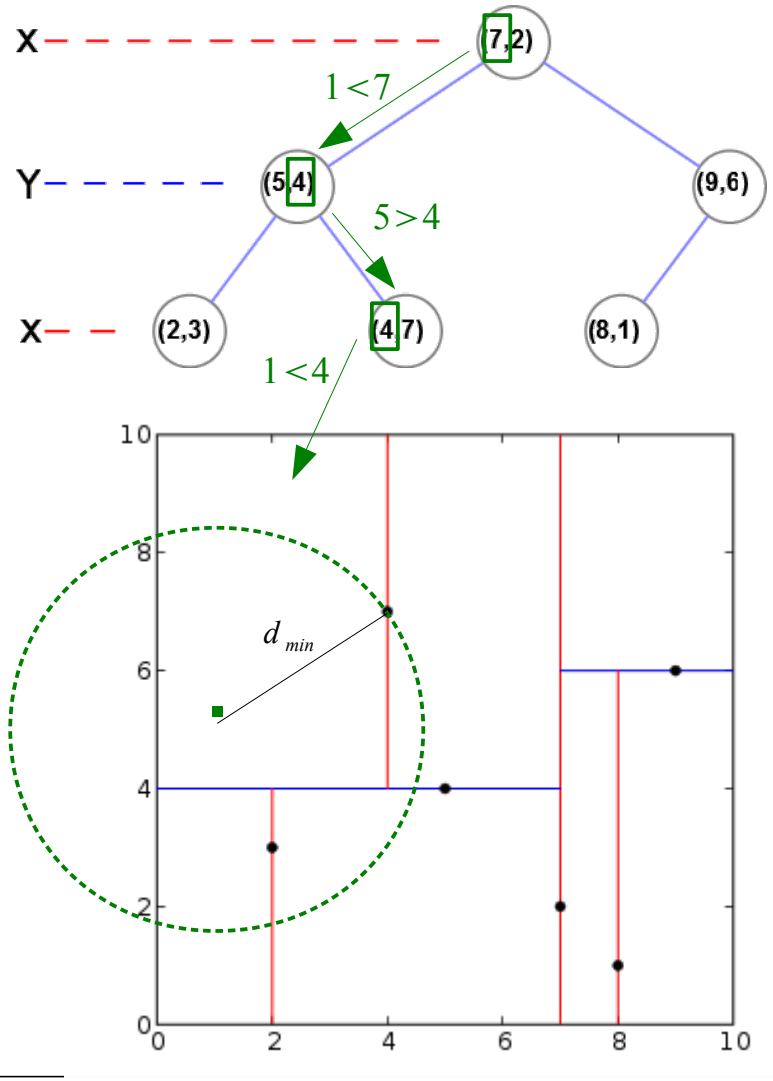
$$d(e_1, e_2) = \sqrt{\sum_A d_A(e_1, e_2)^2}$$

- sort the example down the tree:
  - ends in the left successor of [4,7]
- compute distance to example in the leaf

$$d([1,5],[4,7]) = \sqrt{(1-4)^2 + (5-7)^2} = \sqrt{13}$$

- now we have to look into rectangles that may contain a nearer example
  - remember the difference to the closest example

$$d_{min} = \sqrt{13}$$



# Using kD-trees: example

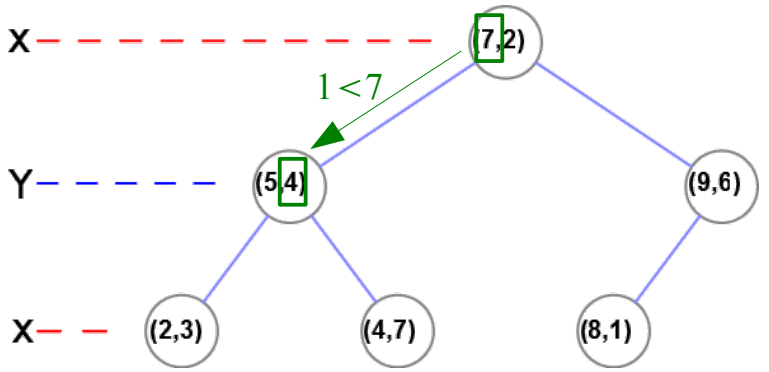
- go up one level (to example [4,7])
- compute distance to the closest point on this split (difference only on X)

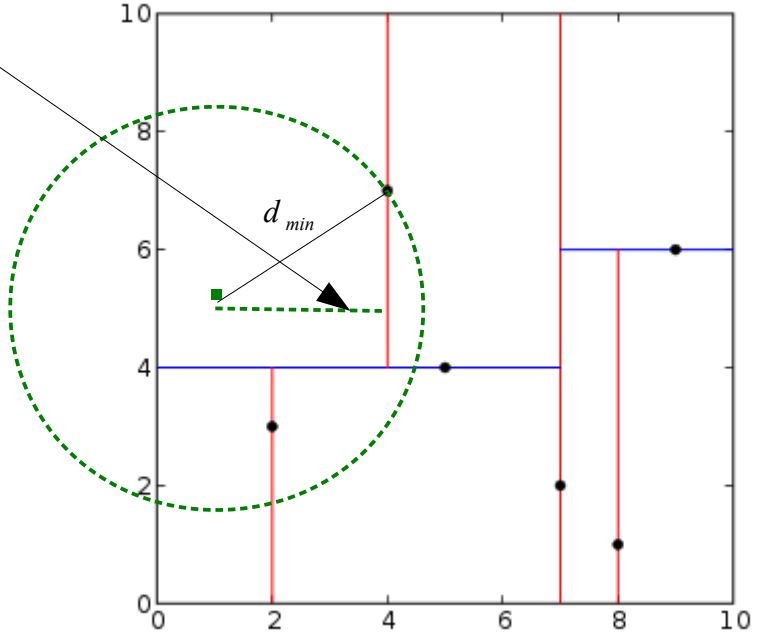
$$d([1,5],[4,*])=\sqrt{(4-1)^2+0^2}=3$$

- If the difference is smaller than the current best difference

$$d([1,5],[4,*])=3<\sqrt{13}=d_{min}$$

- then we could have a closer example in the right subtree of [4,7]
  - which in our case does not contain any example → done



# Using kD-trees: example

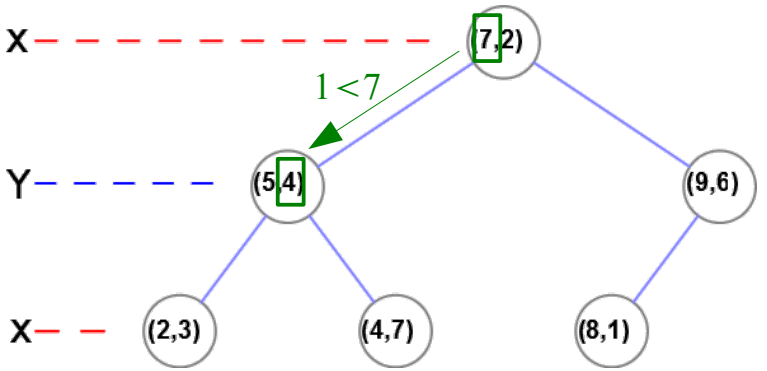
- go up one level (to example [5,4])
- compute distance to the closest point on this split (difference only on Y)

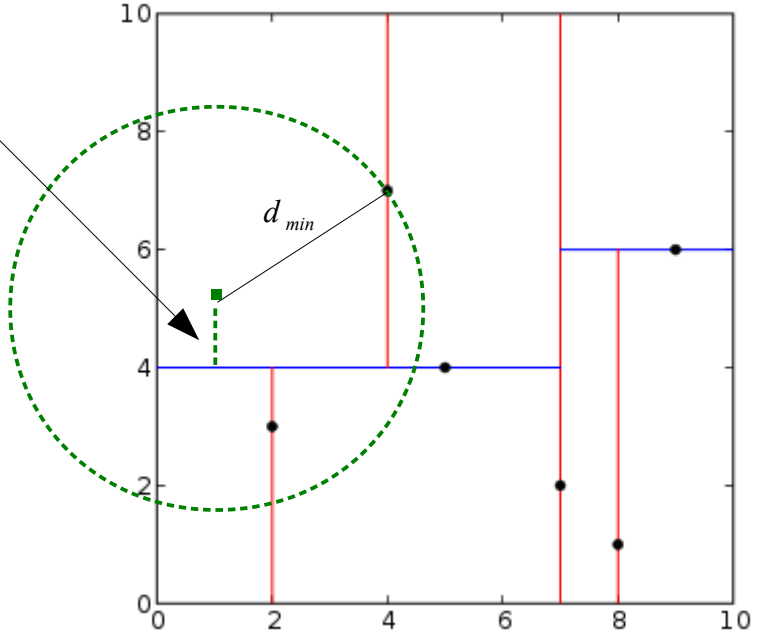
$$d([1,5],[*,4]) = \sqrt{0^2 + (5-4)^2} = 1$$

- if the difference is smaller than the current best difference

$$d([1,5],[*,4]) = 1 < \sqrt{13} = d_{min}$$

- then we could have a closer example in area $Y < 4$.
  - go down the other branch
  - and repeat recursively



# Using kD-trees: example

- go down to leaf [2,3]
- compute distance to example in this leaf

$$d([1,5],[2,3])=\sqrt{(1-2)^2+(5-3)^2}=\sqrt{5}$$

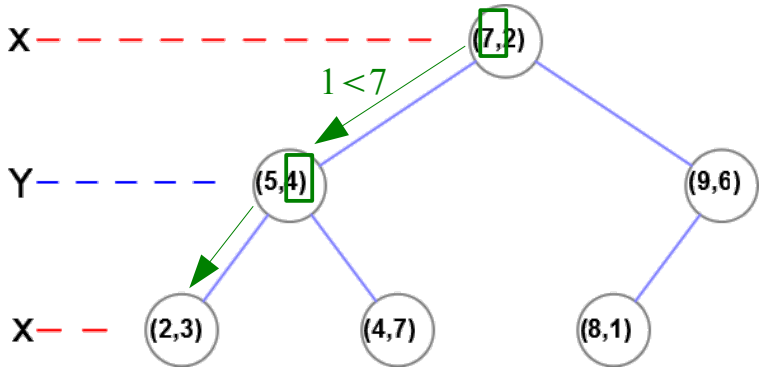
- if the difference is smaller than the current best difference

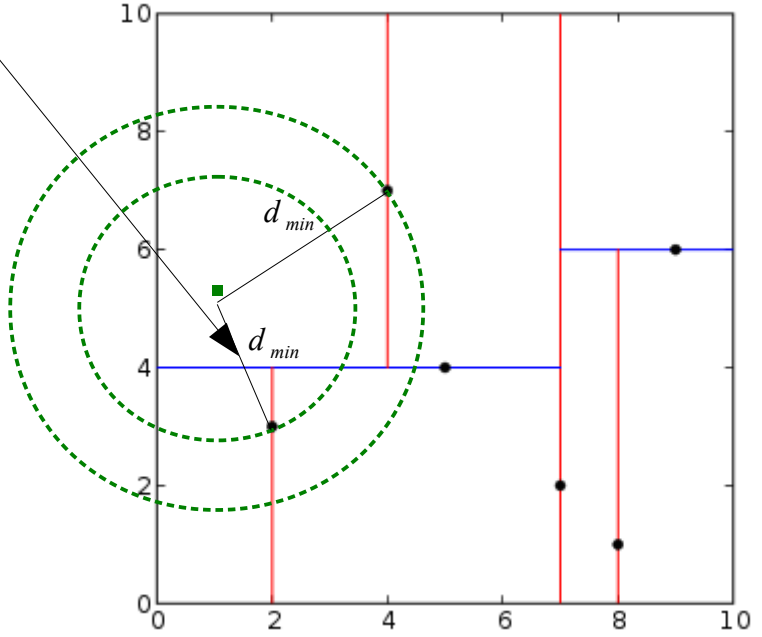
$$d([1,5],[2,3])=\sqrt{5}<\sqrt{13}=d_{min}$$

- then the example in the leaf is the new nearest neighbor and

$$d_{min}=\sqrt{5}<\sqrt{13}$$

- this is recursively repeated until we have processed the root node
  - no more distances have to be computed



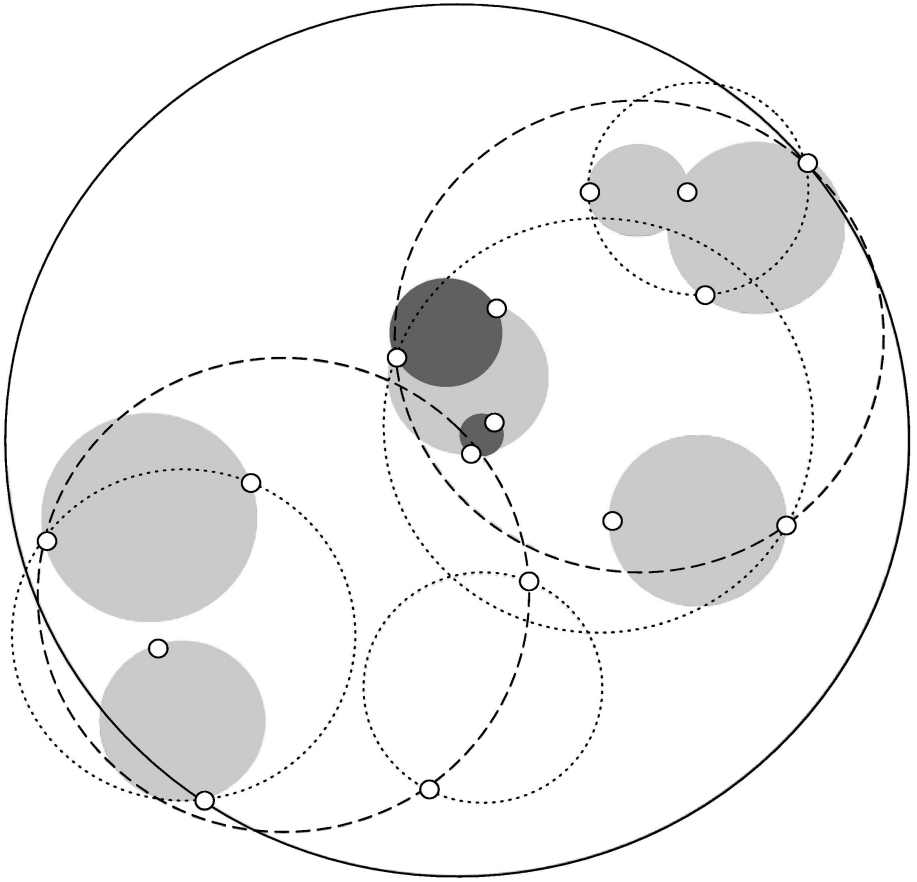
# Ball trees

TECHNISCHE UNIVERSITÄT DARMSTADT

- Problem in kD-trees: corners
- Observation:
  - There is no need to make sure that regions don't overlap

→ We can use balls (hyperspheres) instead of hyperrectangles

  - A ball tree organizes the data into a tree of k-dimensional hyperspheres
  - Normally allows for a better fit to the data and thus more efficient search



# Discussion

TECHNISCHE UNIVERSITÄT DARMSTADT

- Nearest Neighbor methods are often very accurate
  - Assumes all attributes are equally important
    - Remedy: attribute selection or weights
  - Possible remedies against noisy instances
    - Take a majority vote over the *k* nearest neighbors
    - Removing noisy instances from dataset (difficult!)
  - Statisticians have used k-NN since early 1950s
    - If $n \rightarrow \infty$ and $k/n \rightarrow 0$, error approaches minimum
    - can model arbitrary decision boundaries
- ...but somewhat inefficient (at classification time)
  - straight-forward application maybe too slow
  - kD-trees become inefficient when number of attributes is too large (approximately > 10)
  - Ball trees work well in higher-dimensional spaces
- several similarities with rule learning