Applied Machine Learning

Course number: W207

Prof. Alexander I. Iliev, Ph.D.

Applied Machine Learning

Lecture 3 …

- *K-Nearest Neighbors*
- *Probability review*
    - *Conditional probability*
    - *Independence*
- *Naïve Bayes*
    - *Application to text data and spam detection*

Instance-based representation

- Simplest form of learning: *rote learning*
  - Training instances are searched for instance that most closely resembles new instance
  - The instances themselves represent the knowledge
  - Also called *instance-based* learning

- Similarity function defines what's "learned"

- Instance-based learning is *lazy learning*

- Methods: *nearest-neighbor, k-nearest-neighbor, …*

Instance-based representation

- In instance-based classification, each new instance is compared with existing ones using a distance metric
- The closest existing instance is used to assign the class to the new one
- This is called the *nearest-neighbor* classification method
- Sometimes more than one nearest neighbor is used, and the majority class of the closest *k* neighbors (or the distance weighted average if the class is numeric) is assigned to the new instance
- This is the *k-nearest-neighbor* method.

The distance function

- Simplest case: one numeric attribute
  - Distance is the difference between the two attribute values involved (or a function thereof)
- Several numeric attributes: normally, Euclidean distance is used and attributes are normalized
- Nominal attributes: distance is set to 1 if values are different, 0 if they are equal
- Are all attributes equally important?
  - Weighting the attributes might be necessary

Instance-based learning

- In instance-based learning the distance function defines what is learned

- Most instance-based schemes use *Euclidean distance*:

$$\sqrt{(a_1^{(1)} - a_1^{(2)})^2 + (a_2^{(1)} - a_2^{(2)})^2 + \cdots + (a_k^{(1)} - a_k^{(2)})^2}$$

$\mathbf{a}^{(1)}$ and $\mathbf{a}^{(2)}$: two instances with k attributes

- **Note**: taking the square root is not required when comparing distances

- Other popular metric: *city-block metric (aka Manhattan)*
  - Adds differences without squaring them

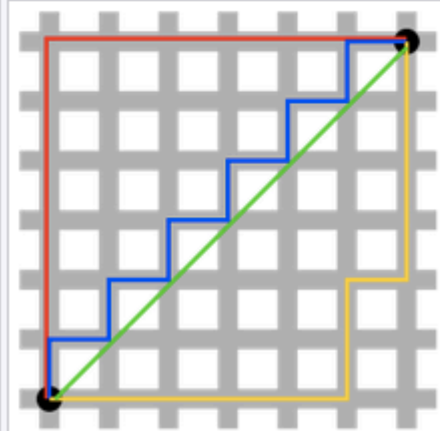
Manhattan Distance

Taxicab geometry

From Wikipedia, the free encyclopedia
(Redirected from Manhattan distance)

A **taxicab geometry**, considered by Hermann Minkowski in 19th-century Germany, is a form of geometry in which the usual distance function or metric of Euclidean geometry is replaced by a new metric in which the distance between two points is the sum of the absolute differences of their Cartesian coordinates.

The **taxicab metric** is also known as **rectilinear distance**, ***L_1* distance** or **ℓ_1 norm** (see L^p space), **snake distance**, **city block distance**, **Manhattan distance** or **Manhattan length**, with corresponding variations in the name of the geometry.[1] The latter names allude to the grid layout of most streets on the island of Manhattan, which causes the shortest path a car could take between two intersections in the borough to have length equal to the intersections' distance in taxicab geometry.

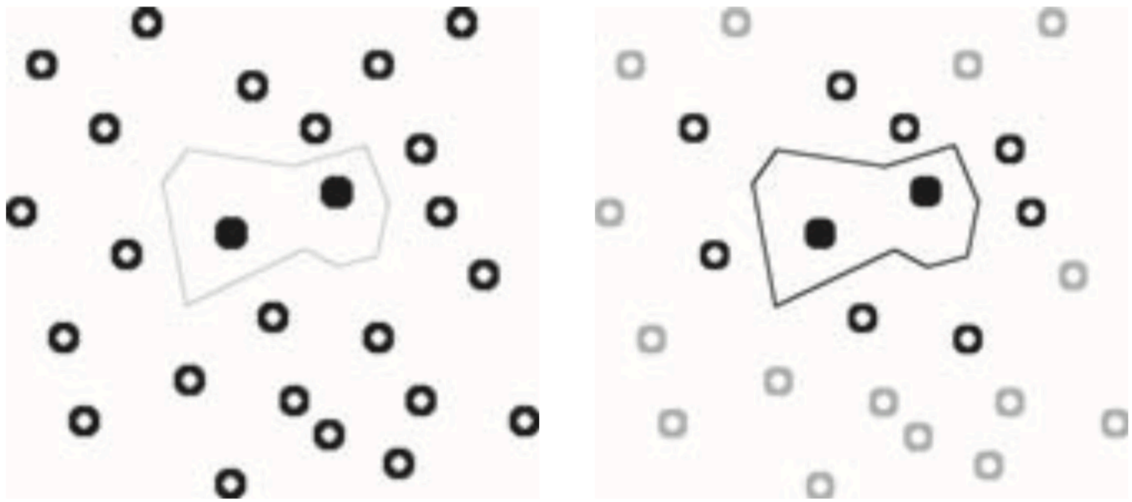


Taxicab geometry versus Euclidean distance: In taxicab geometry, the red, yellow, and blue paths all have the shortest length of 12. In Euclidean geometry, the green line has length $6\sqrt{2} \approx 8.49$, and is the unique shortest path.

Contents [hide]

Source: Wikipedia

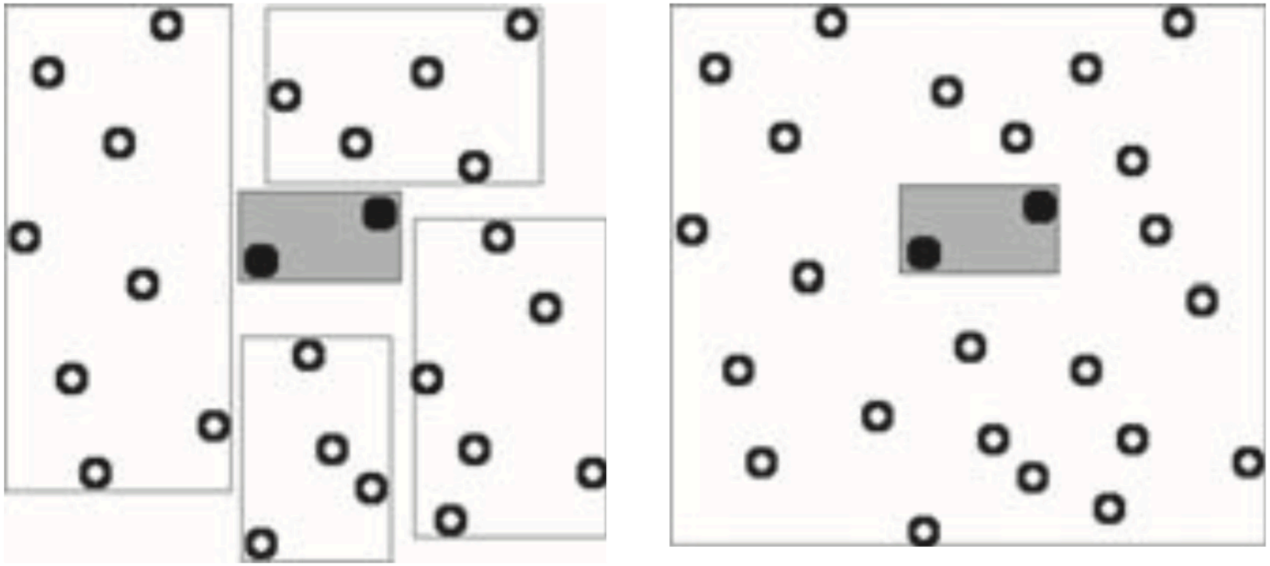
Learning prototypes



(a) (b)

- Only those instances involved in a decision need to be stored
- Noisy instances should be filtered out
- Idea: only use *prototypical* examples

Rectangular generalizations



(c) (d)

- Nearest-neighbor rule is used outside rectangles
- Rectangles are rules! (But they can be more conservative than "normal" rules.)
- Nested rectangles are rules with exceptions

Finding nearest neighbors efficiently

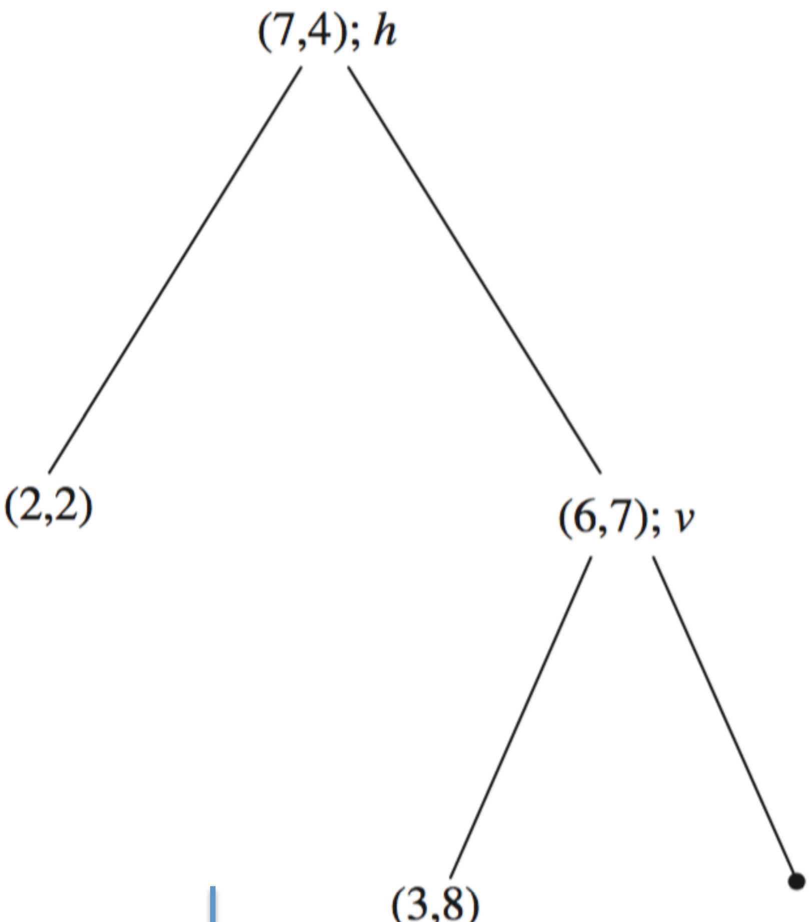
- Simplest way of finding nearest neighbor:
  linear scan of the data
  - Classification takes time proportional to the product of the number of instances in training and test sets

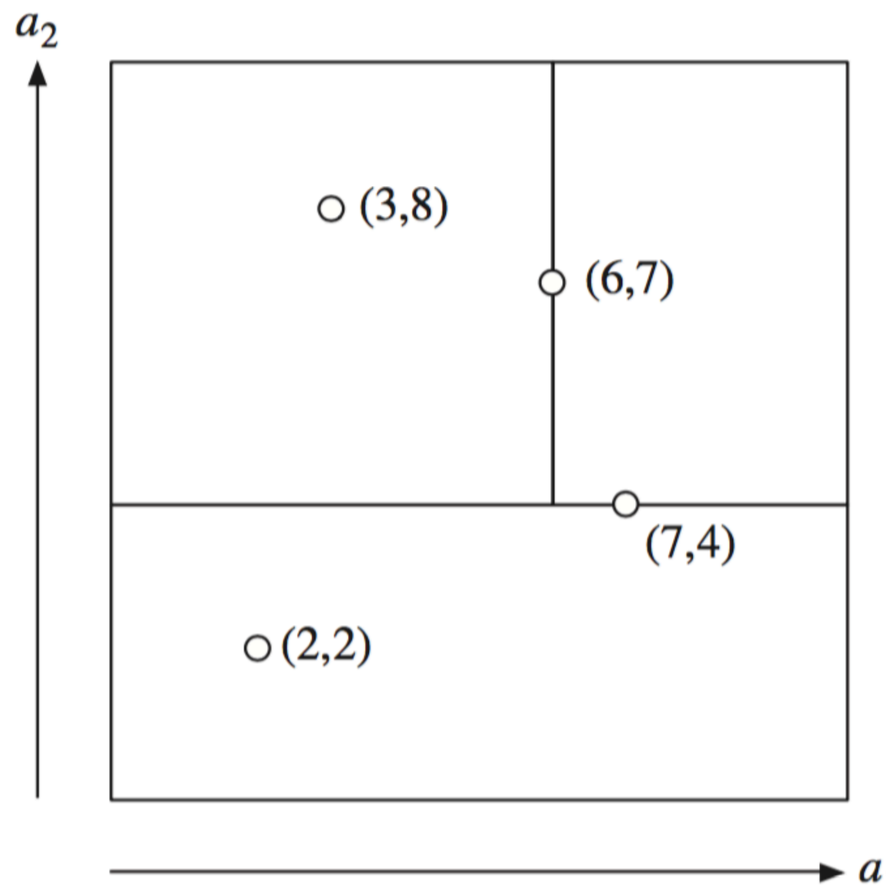
- Nearest-neighbor search can be done more efficiently using appropriate data structures

- Two methods that represent training data in a tree structure are:

  kD-trees and *ball trees*

kD-tree example





Shows an example with $k = 2$

Shows the four training instances it represents, along with the hyperplanes that constitute the tree

Discussion of nearest-neighbor learning

- Often very accurate
- **Assumes** all attributes are equally important
  - Remedy: attribute selection, attribute weights, or attribute scaling
- **Possible remedies** against noisy instances:
  - Take a majority vote over the *k* nearest neighbors
  - Remove noisy instances from dataset (difficult!)
- Statisticians have used *k*-NN since the early 1950s
  - If $n \to \infty$ and $k/n \to 0$, classification error approaches minimum
- *k*D-trees can become inefficient when the number of attributes is too large
- Ball trees are instances *may* help; they are instances of *metric trees*

Model selection criteria

- Model selection criteria attempt to find a good compromise between:
    - The complexity of a model
    - Its prediction accuracy on the training data
- Reasoning: a good model is a simple model that achieves high accuracy on the given data
- Also known as *Occam's Razor*:

  "*the best theory is the smallest one that describes all the facts*"

  William of Ockham, born in the village of Ockham in Surrey (England) about 1285, was the most influential philosopher of the 14th century and a controversial theologian.

Elegance vs. errors

- Theory 1: very simple, elegant theory that explains the data almost perfectly
- Theory 2: significantly more complex theory that reproduces the data without mistakes … *(is it more elegant?)*
- Theory 1 is probably preferable
- Classical example: Kepler's three laws on planetary motion
  - Less accurate than Copernicus's latest refinement of the Ptolemaic theory of epicycles on the data available at the time

Simplicity first

- Simple algorithms often work very well!
- There are many kinds of simple structure, e.g.:
  - One attribute does all the work
  - All attributes contribute equally & independently
  - Logical structure with a few attributes suitable for tree
  - An independent set of simple logical rules
  - Relationships between groups of attributes
  - A weighted linear combination of the attributes
  - Strong neighborhood relationships based on distance
  - Clusters of data in unlabeled data
  - Bags of instances that can be aggregated
- Success of method depends on the domain

Applied Machine Learning

Lecture 3 …

- *K-Nearest Neighbors*
- *Probability review*
    - *Conditional probability*
    - *Independence*
- *Naïve Bayes*
    - *Application to text data and spam detection*

Applied Machine Learning

Lecture 3 …

- *K-Nearest Neighbors*
- *Probability review*
    - *Conditional probability*
    - *Independence*
- *Naïve Bayes*
    - *Application to text data and spam detection*