Machine Learning and Pervasive Computing

Stephan Sigg

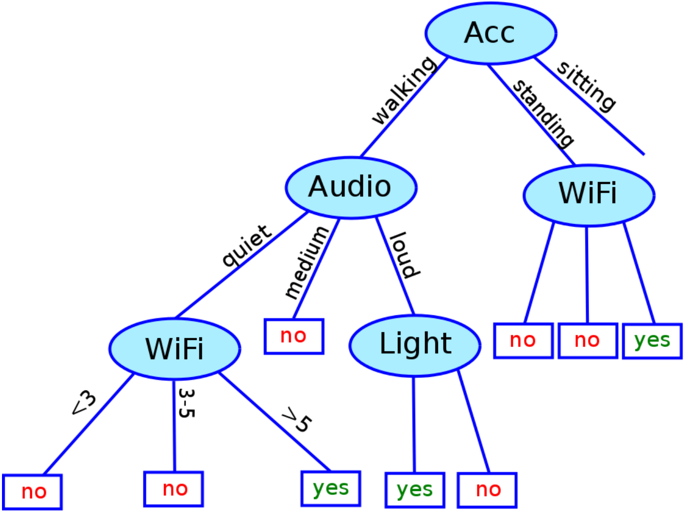
Georg-August-University Goettingen, Computer Networks

15.06.2015

Overview and Structure

13.04.2015 Organisation
13.04.2015 Introduction
20.04.2015 Rule-based learning
27.04.2015 Decision Trees
04.05.2015 A simple Supervised learning algorithm
11.05.2015 –
18.05.2015 Excursion: Avoiding local optima with random search
25.05.2015 –
01.06.2015 High dimensional data
08.06.2015 Artificial Neural Networks
15.06.2015 k-Nearest Neighbour methods
22.06.2015 Probabilistic models
29.06.2015 Topic models
06.07.2015 Unsupervised learning
13.07.2015 Anomaly detection, Online learning, Recom. systems

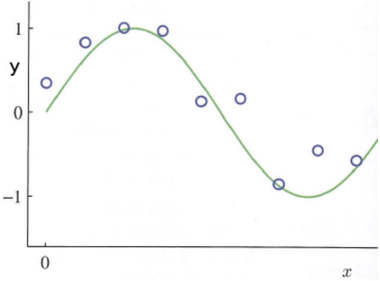
Recap: Decision tree



Recap: Polynomial curve fitting

Fit data points to a polynomial function:

$$h(x, \vec{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

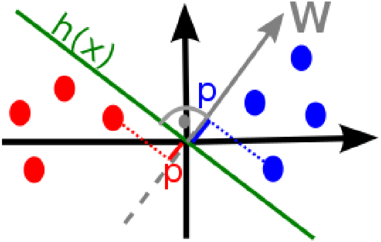


Recap: Support vector machines (SVM)

$$\min_W \quad \frac{1}{2}\sum_{j=1}^n w_j^2 = \frac{1}{2}\left(\sqrt{w_1^2+\cdots+w_n^2}\right)^2 \quad = \frac{1}{2}||W||^2$$

$$s.t. \quad W^Tx_i \geq 1 \text{ if } y_i = 1 \quad \rightarrow ||W|| \cdot p_i \geq 1$$

$$W^Tx_i \leq -1 \text{ if } y_i = 0 \quad \rightarrow ||W|| \cdot p_i \leq -1$$



Which decision boundaray is found?

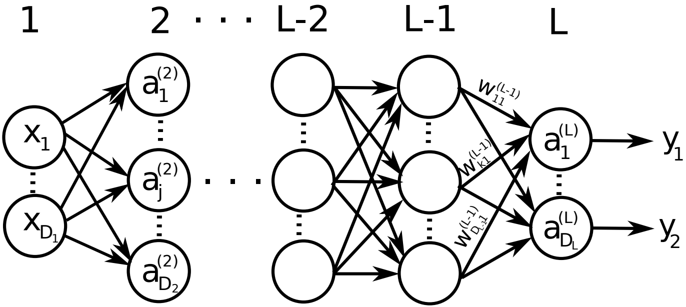
$$h(x) = w_1x_1 + w_2x_2$$

$\rightarrow$ W orthogonal to all x with $h(x) = 0$

$\Rightarrow$ min $\frac{1}{2}||W||^2$ and $||W|| \cdot p_i \geq 1$ necessitate larger p_i

Recap: Neural networks

$$h_k(\vec{x}, \vec{w}) = f_{\text{act}}^{(3)} \left(\sum_{j=1}^{D_2} w_{jk}^{(2)} f_{\text{act}}^{(2)} \left(\sum_{i=1}^{D_1} w_{ij}^{(1)} x_i + w_{0j}^{(1)} \right) + w_{0k}^{(2)} \right)$$



Nearest neighbour methods

Instance-based learning

In instance-based learning, classification is not derived from rules or functions but from the instances themselves

Outline

mvote.ugoe.de/281D

Histogram methods

Parzen Estimator methods

Nearest neighbour techniques
- Distance calculation
- kD-trees
- Ball trees

Histogram methods

mvote.ugoe.de/281D

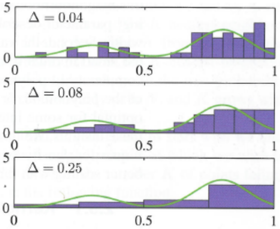
Alternative approach to function estimation: histogram methods

In general, the probability density of an event is estimated by dividing the range of N values into bins of size Δ_i

Then, count the number of observations that fall inside bin Δ_i

This is expressed as a normalised probability density

$$p_i = \frac{n_i}{N\Delta_i}$$

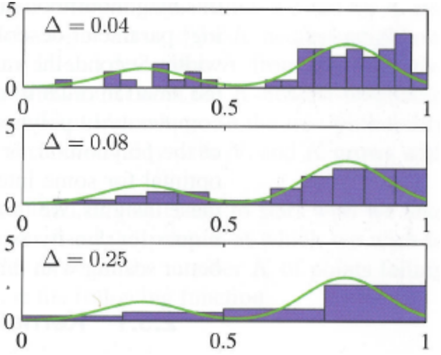


Histogram methods

mvote.ugoe.de/281D

Accuracy of the estimation is dependent on the width of the bins

Approach well suited for big data since the data items can be discarded once the histogram is created



mvote.ugoe.de/281E

Outline

Histogram methods

Parzen Estimator methods

Nearest neighbour techniques
- Distance calculation
- kD-trees
- Ball trees

mvote.ugoe.de/281E

Parzen estimator methods

Assume an unknown probability density $\mathcal{P}(\cdot)$

Considering a small region $\mathcal{R}$ around $\overrightarrow{x}$:

$$P = \int_{\mathcal{R}} \mathcal{P}(\overrightarrow{x}) d\overrightarrow{x}$$

mvote.ugoe.de/281E

Parzen estimator methods

With the binomial distribution we can express the probability that K out of N ponts fall into $\mathcal{R}$:

$$\begin{aligned}\mathrm{Bin}(K|N,P) &= \binom{N}{K} P^K (1-P)^{N-K} \\ &= \frac{N!}{K!(N-K)!} P^K (1-P)^{N-K}\end{aligned}$$

mvote.ugoe.de/281E

Parzen estimator methods

For large N we can show

$$K \approx NP$$

mvote.ugoe.de/281E

Parzen estimator methods

For large N we can show

$$K \approx NP$$

With sufficiently small $\mathcal{R}$ we can also show for the volume V of $\mathcal{R}$

$$P \approx \mathcal{P}(\vec{x})V$$

mvote.ugoe.de/281E

Parzen estimator methods

For large N we can show

$$K \approx NP$$

With sufficiently small $\mathcal{R}$ we can also show for the volume V of $\mathcal{R}$

$$P \approx \mathcal{P}(\overrightarrow{x})V$$

Therefore, we can estimate the density as

$$\mathcal{P}(\overrightarrow{x}) = \frac{K}{NV}$$

Parzen estimator methods

We assume that $\mathcal{R}$ is a small hypercube

In order to count the number K of points that fall inside $\mathcal{R}$ we define

$$k(\overrightarrow{u}) = \begin{cases} 1, & ||u_i|| \leq \frac{1}{2}, \quad i = 1, \dots, D, \\ 0, & \text{otherwise} \end{cases}$$

This represents a sphere with diameter 1 centred around the origin

This function is an example of a Parzen window

mvote.ugoe.de/281E

Parzen estimator methods

$$k(\overrightarrow{u}) = \begin{cases} 1, & |u_i| \leq \frac{1}{2}, \quad i = 1, \ldots, D, \\ 0, & \text{otherwise} \end{cases}$$

When the measured data point $\overrightarrow{x_n}$ lies inside a sphere of radius h centred around $\overrightarrow{x}$, we have

$$k\left(\frac{\overrightarrow{x} - \overrightarrow{x_n}}{h}\right) = 1$$

The total count K of points that fall inside this sphere is

$$K = \sum_{n=1}^{N} k\left(\frac{\overrightarrow{x} - \overrightarrow{x_n}}{h}\right)$$

mvote.ugoe.de/281E

Parzen estimator methods

The total count K of points that fall inside this cube is

$$K = \sum_{n=1}^{N} k\left(\frac{\overrightarrow{x} - \overrightarrow{x_n}}{h}\right)$$

When we substitute this in the density estimate derived above

$$\mathcal{P}(\overrightarrow{x}) = \frac{K}{NV}$$

with volume $V = h^D$ we obtain the overall density estimate as

$$\mathcal{P}(\overrightarrow{x}) = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{h^D}\left(\frac{\overrightarrow{x} - \overrightarrow{x_n}}{h}\right)$$

Parzen estimator methods

mvote.ugoe.de/281E

$$\mathcal{P}(\vec{x}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{h^D}\left(\frac{\vec{x}-\vec{x_n}}{h}\right)$$

Again, this density estimator suffers from artificial discontinuities

(Due to the fixed boundaries of the sphere)

Problem can be overcome by choosing a smoother kernel function

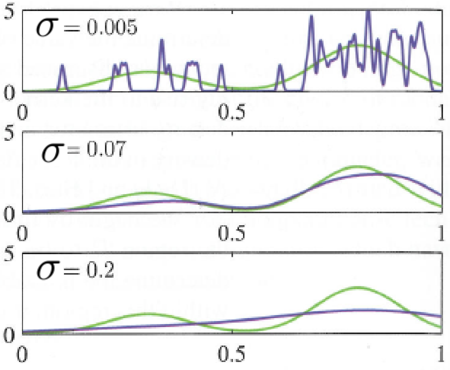
(A common choice is a Gaussian kernel with a standard deviation σ)

$$\mathcal{P}(\vec{x}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{(2\pi\sigma^2)^{\frac{D}{2}}}e^{-\frac{||\vec{x}-\vec{x_n}||^2}{2\sigma^2}}$$

mvote.ugoe.de/281E

Parzen estimator methods

Density estimation for various values of σ



Outline

mvote.ugoe.de/281F

Histogram methods

Parzen Estimator methods

Nearest neighbour techniques
- Distance calculation
- kD-trees
- Ball trees

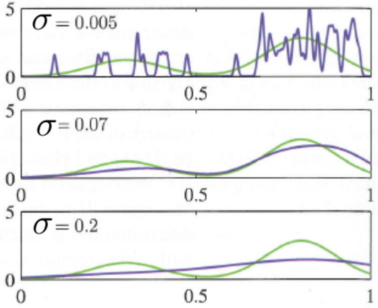
mvote.ugoe.de/281F

Nearest neighbour methods

A problem with Parzen estimator methods is that the parameter governing the kernel width (h or σ) is fixed for all values $\vec{x}$

In regions with

...high density, a wide kernel might lead to over-smoothing
...low density, the same width may lead to noisy estimates



Nearest neighbour methods

mvote.ugoe.de/281F

NN-methods address this by adapting width to data density

Parzen estimator methods fix V and determine K from the data
Nearest neighbour methods fix K and choose V accordingly

Again, we consider a point $\overrightarrow{x}$ and estimate the density $\mathcal{P}(\overrightarrow{x})$

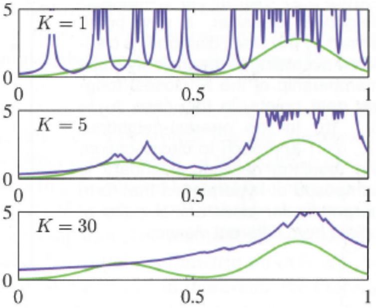
The radius of the sphere is increased until K data points (the nearest neighbours) are covered

Nearest neighbour methods

mvote.ugoe.de/281F

The value K then controls the amount of smoothing

Again, an optimum value for K exists



Recap: Conditional probability

Conditional probability

The conditional probability of two events χ_1 and χ_2 with $P(\chi_2) > 0$ is denoted by $P(\chi_1|\chi_2)$ and is calculated by

$$P(\chi_1|\chi_2) = \frac{P(\chi_1 \cap \chi_2)}{P(\chi_2)}$$

$P(\chi_1|\chi_2)$ describes the probability that event χ_2 occurs in the presence of event χ_2.

mvote.ugoe.de/281F

Recap: Bayes rule

With the notion of conditional probability we can express the effect of observed data $\overrightarrow{a} = a_1, \ldots, a_N$ on a probability distribution of $\overrightarrow{b}$: $P(\overrightarrow{b})$.

Thomas Bayes described a way to evaluate the uncertainty of $\overrightarrow{b}$ after observing $\overrightarrow{a}$

$$P(\overrightarrow{b}|\overrightarrow{a}) = \frac{P(\overrightarrow{a}|\overrightarrow{b})P(\overrightarrow{b})}{P(\overrightarrow{a})}$$

$P(\overrightarrow{a}|\overrightarrow{b})$ expresses how probable a value for $\overrightarrow{a}$ is given a fixed choice of $\overrightarrow{b}$

Nearest neighbour methods

mvote.ugoe.de/281F

Classification: Apply KNN-density estimation for each class

Assume data set of N points with N_k points in class C_k

To classify sample $\vec{x}$, draw a sphere containing K points around $\vec{x}$

Sphere can contain other points regardless of their class

Assume sphere has volume V and contains K_k points from C_k

mvote.ugoe.de/281F

Nearest neighbour methods

Assume: Sphere of volume V contains K_k points from class C_k

We estimate the density of class C_k as

$$\mathcal{P}(\vec{x}|C_k) = \frac{K_k}{N_k V}$$

The unconditional density is given as

$$\mathcal{P}(\vec{x}) = \frac{K}{NV}$$

The probability to experience a class C_k is given as

$$\mathcal{P}(C_k) = \frac{N_k}{N}$$

mvote.ugoe.de/281F

Nearest neighbour methods

Assume: Sphere of volume V contains K_k points from class C_k

We estimate the density of class C_k as

$$\mathcal{P}(\vec{x}|C_k) = \frac{K_k}{N_k V}$$

The unconditional density is given as

$$\mathcal{P}(\vec{x}) = \frac{K}{NV}$$

The probability to experience a class C_k is given as

$$\mathcal{P}(C_k) = \frac{N_k}{N}$$

With Bayes theorem we can combine this to achieve

$$\mathcal{P}(C_k|\vec{x}) = \frac{\mathcal{P}(\vec{x}|C_k)\mathcal{P}(C_k)}{\mathcal{P}(\vec{x})} = \frac{K_k}{K}$$

Nearest neighbour methods

mvote.ugoe.de/281F

$$\mathcal{P}(C_k|\overrightarrow{x}) = \frac{\mathcal{P}(\overrightarrow{x}|C_k)\mathcal{P}(C_k)}{\mathcal{P}(\overrightarrow{x})} = \frac{K_k}{K}$$

To minimise the probability of misclassification, assign $\overrightarrow{x}$ to class with the largest probability

This corresponds to the largest value of

$$\frac{K_k}{K}$$

Nearest neighbour methods

mvote.ugoe.de/281F

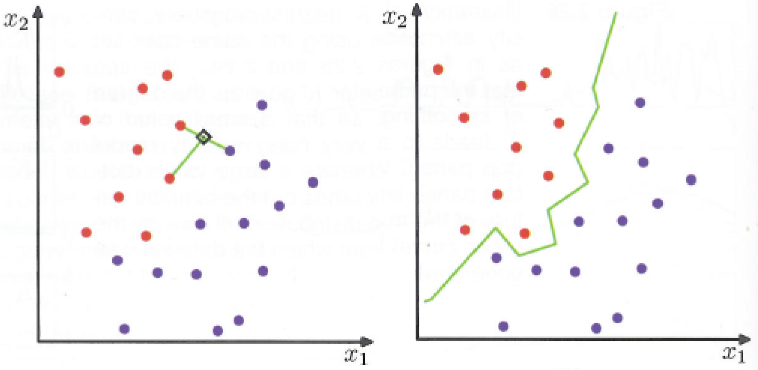
To classify a point, we identify the K nearest points

And assign the point to the class having most representatives in this set

mvote.ugoe.de/281F

Nearest neighbour methods

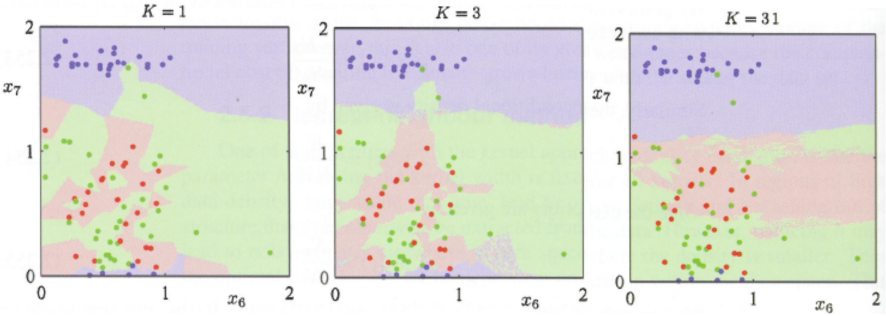
Classification of points by the K-nearest neighbour classifier



Nearest neighbour methods

mvote.ugoe.de/281F

Classification of points by the K-nearest neighbour classifier



Nearest neighbour methods

mvote.ugoe.de/281F

Instance-based learning

In instance-based learning, classification is not derived from rules or functions but from the instances themselves

Nearest neighbour methods

mvote.ugoe.de/281F

Instance-based learning

In instance-based learning, classification is not derived from rules or functions but from the instances themselves

Nearest neighbour classification New instances are compared with existing ones and nearest neighbours are used to predict a class

mvote.ugoe.de/281F

Nearest neighbour methods

Instance-based learning

In instance-based learning, classification is not derived from rules or functions but from the instances themselves

Nearest neighbour classification New instances are compared with existing ones and nearest neighbours are used to predict a class

k-nearest neighbour Majority vote among k nearest neighbours

mvote.ugoe.de/281F

Nearest neighbour methods

Storage demands KNN and Parzen-method are not well suited for large data sets since they require the entire data set to be stored

mvote.ugoe.de/281F

Nearest neighbour methods

Storage demands KNN and Parzen-method are not well suited for large data sets since they require the entire data set to be stored

Instance-based learning Distance function to determine which member of a training set is closest to an unknown test instance

⇒ How to calculate the distance?

mvote.ugoe.de/281F

Nearest neighbour methods

Storage demands KNN and Parzen-method are not well suited for large data sets since they require the entire data set to be stored

Instance-based learning Distance function to determine which member of a training set is closest to an unknown test instance

$\Rightarrow$ How to calculate the distance?

Low Classification speed The intuitive way to find nearest neighbours involves linear comparison to all training examples

$\Rightarrow$ Can we store and process data more efficiently?

Nearest neighbour methods

mvote.ugoe.de/281F

Storage demands KNN and Parzen-method are not well suited for large data sets since they require the entire data set to be stored

Instance-based learning Distance function to determine which member of a training set is closest to an unknown test instance

$\Rightarrow$ How to calculate the distance?

Low Classification speed The intuitive way to find nearest neighbours involves linear comparison to all training examples

$\Rightarrow$ Can we store and process data more efficiently?

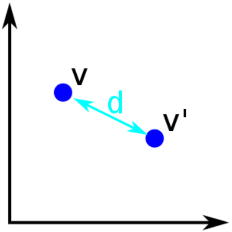
mvote.ugoe.de/281F

Nearest neighbour methods

Distance function

Most instance-based learners utilise Euclidean distance:

$$\sqrt{(v_1 - v_1')^2 + (v_2 - v_2')^2 + \cdots + (v_k - v_k')^2}$$



mvote.ugoe.de/281F

Nearest neighbour methods

Distance function

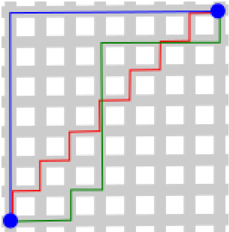
Most instance-based learners utilise Euclidean distance:

$$\sqrt{(v_1 - v_1')^2 + (v_2 - v_2')^2 + \cdots + (v_k - v_k')^2}$$

Alternatives:

- Manhattan Distance

$$(v_1 - v_1') + (v_2 - v_2') + \cdots + (v_k - v_k')$$



mvote.ugoe.de/281F

Nearest neighbour methods

Distance function

Most instance-based learners utilise Euclidean distance:

$$\sqrt{(v_1 - v_1')^2 + (v_2 - v_2')^2 + \cdots + (v_k - v_k')^2}$$

Alternatives:

- Manhattan Distance

$$(v_1 - v_1') + (v_2 - v_2') + \cdots + (v_k - v_k')$$

- Powers higher than square
  Increase the influence of large differences at the expense of small differences

It is important to think of actual instances and what it means for them to be separated by a certain distance

Nearest neighbour methods
Different scales

mvote.ugoe.de/281F

Different features often follow different scales.

Normalize! It is usually a good idea to normalize all features first

$$\text{feature}_i = \frac{v_i - \min(v_i)}{\max(v_i) - \min(v_i)}$$

mvote.ugoe.de/281F

Nearest neighbour methods

Nominal features

Nominal features that take symbolic rather than numeric values, have to be handled differently

mvote.ugoe.de/281F

Nearest neighbour methods

Nominal features

Nominal features that take symbolic rather than numeric values, have to be handled differently

Common solution :

Features with identical value Distance $= 0$
Features with different value Distance $= 1$

More expressive metric: e.g. hue in color space for colors

mvote.ugoe.de/281F

Nearest neighbour methods
Missing feature values

For missing feature values, commonly the distance is chosen as large as possible

(if both are missing, Distance= 1, if only one is missing:)

mvote.ugoe.de/281F

Nearest neighbour methods
Missing feature values

For missing feature values, commonly the distance is chosen as large as possible

(if both are missing, Distance$=1$, if only one is missing:)

Nominal features Distance $= 1$

Numeric features Distance $= \max(v', 1 - v')$ (where v' is the (normalized) value to compare to)

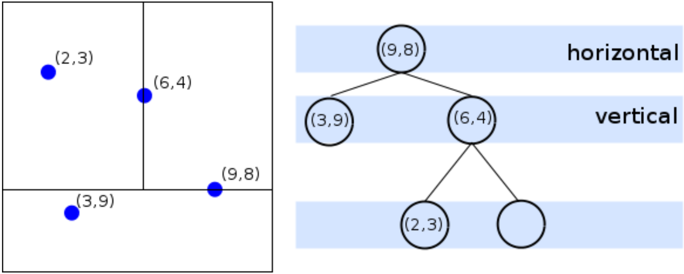
mvote.ugoe.de/281F

Nearest neighbour methods

Finding nearest neighbours efficiently

Instance-based learning is often slow

- Intuitive way to find nearest neighbour is to iteratively compare to all training examples

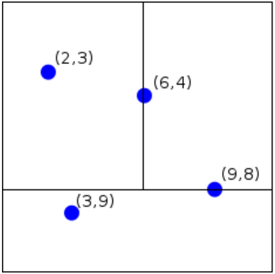


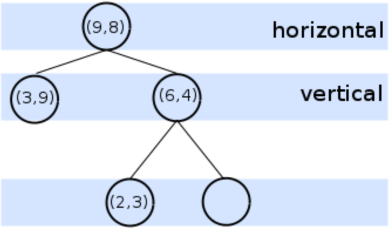
mvote.ugoe.de/281F

Nearest neighbour methods

Finding nearest neighbours efficiently

- Instance-based learning is often slow
  - Intuitive way to find nearest neighbour is to iteratively compare to all training examples
- More efficient search for nearest neighbour possible by kD-tree
  - Binary tree that stores points from k-dimensional space



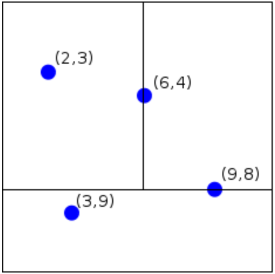


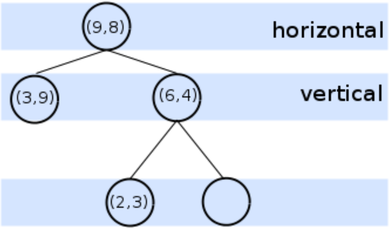
mvote.ugoe.de/281F

Nearest neighbour methods

kD-Trees

→ Each region contains 0-1 points
→ Every point in the training set correpsonds to a sigle node
→ up to half the nodes are at the leaves of the tree

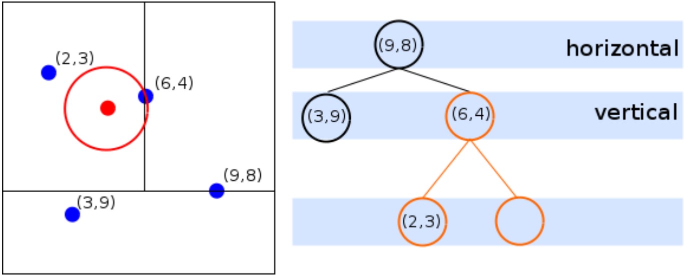




mvote.ugoe.de/281F

Nearest neighbour methods

Find nearest Neighbour in a kD-Tree :

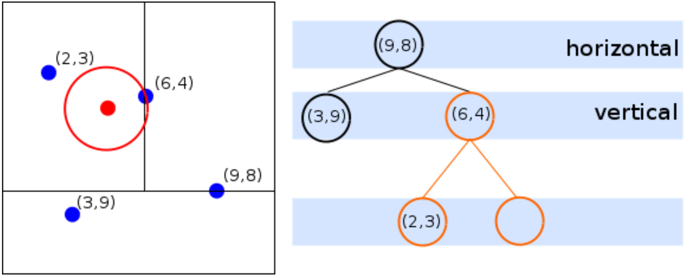


mvote.ugoe.de/281F

Nearest neighbour methods

Find nearest Neighbour in a kD-Tree :

1. Treverse the nodes of the tree to locate the region containing the point
2. Check parent nodes and other siblings of the parent for their distance to the point
   - might be necessary to recursively repeat for parent node



mvote.ugoe.de/281F

Nearest neighbour methods

Creating good kD-Trees

Unbalanced trees lead to low efficiency gain

Backtracking cost is lowest for approximately square regions

mvote.ugoe.de/281F

Nearest neighbour methods

Creating good kD-Trees

1. How to select good first instance to split on ?
2. How to determine dimension of split ?

mvote.ugoe.de/281F

Nearest neighbour methods

Creating good kD-Trees

1. How to select good first instance to split on ?
2. How to determine dimension of split ?

Find dimension for a split

- Calculate variance of data points along each axis individually
- Select axis with greatest variance and create hyperplane perpendicular to it
  - Split perpendicular to direction of greatest spread

mvote.ugoe.de/281F

Nearest neighbour methods

Creating good kD-Trees

1. How to select good first instance to split on ?
2. How to determine dimension of split ?

Find dimension for a split

Calculate variance of data points along each axis individually

Select axis with greatest variance and create hyperplane perpendicular to it

Split perpendicular to direction of greatest spread

Find good first instance for a split

Calculate median along that axis and select the corresponding point

Half of the points lie on either side of the split

Nearest neighbour methods

kD-trees – online updates

- On-line learning with kD-trees is possible by appending new samples to an existing tree

mvote.ugoe.de/281F

Nearest neighbour methods

kD-trees – online updates

- On-line learning with kD-trees is possible by appending new samples to an existing tree

	Traverse each new sample down to a leaf of an existing tree to find its hyperrectangle
If empty	place new point there
else	Split hyperrectangle along its longest dimension

mvote.ugoe.de/281F

Nearest neighbour methods

kD-trees – high dimensions

A problem with kD-trees

- Especially in higher dimensions:
- Corners of rectangles may contain points farther to the center of the enclosing rectangle than to other rectangles
- Then, unnecessarily many branches of the tree are considered which reduces efficiency

Nearest neighbour methods

Ball trees

mvote.ugoe.de/281F

Ball trees

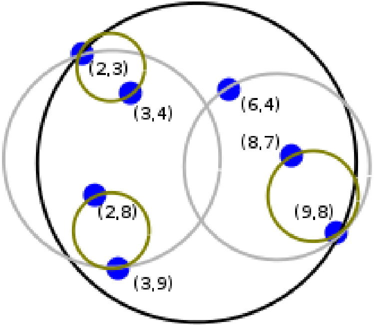
- Use hyperspheres not hyperrectangles
- Binary tree
- Each node defines the center and radius of the smallest hypersphere that contains all nodes of its sub-trees

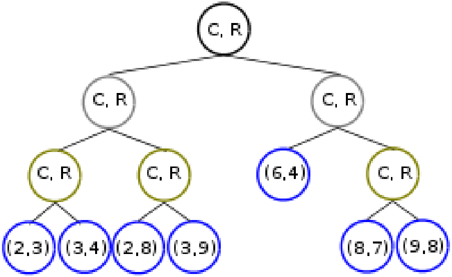
Recursive construction: Split data into two sets along the dimension with greatest spread (split at median point)

Nearest neighbour methods

Ball trees

mvote.ugoe.de/281F





Outline

Histogram methods

Parzen Estimator methods

Nearest neighbour techniques
- Distance calculation
- kD-trees
- Ball trees

Questions?

Stephan Sigg
stephan.sigg@cs.uni-goettingen.de

Literature

- C.M. Bishop: Pattern recognition and machine learning, Springer, 2007.
- R.O. Duda, P.E. Hart, D.G. Stork: Pattern Classification, Wiley, 2001.