Workshop on belief functions

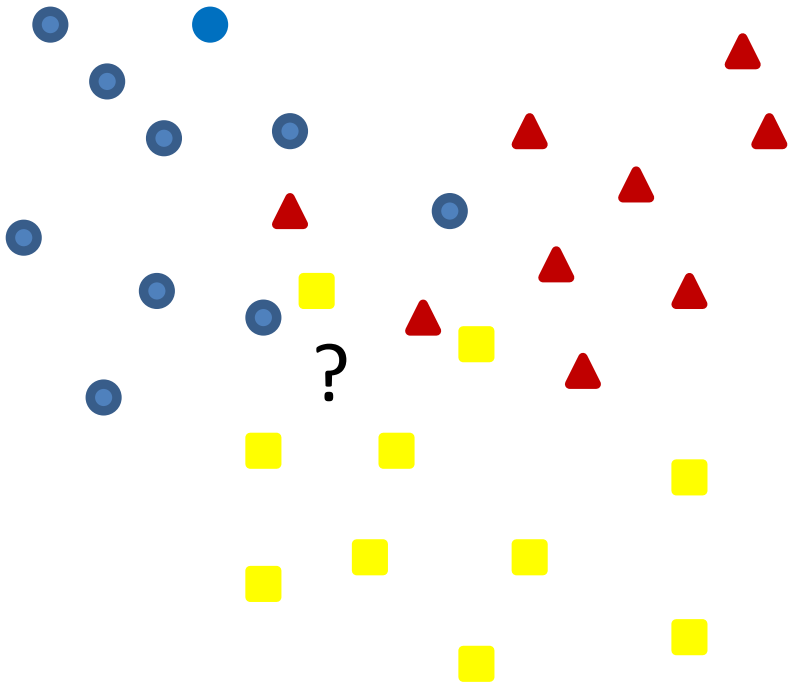
Classification

Thierry Denœux

Université de Technologie de Compiègne, France
HEUDIASYC (UMR CNRS 7253)
https://www.hds.utc.fr/~tdenoeux

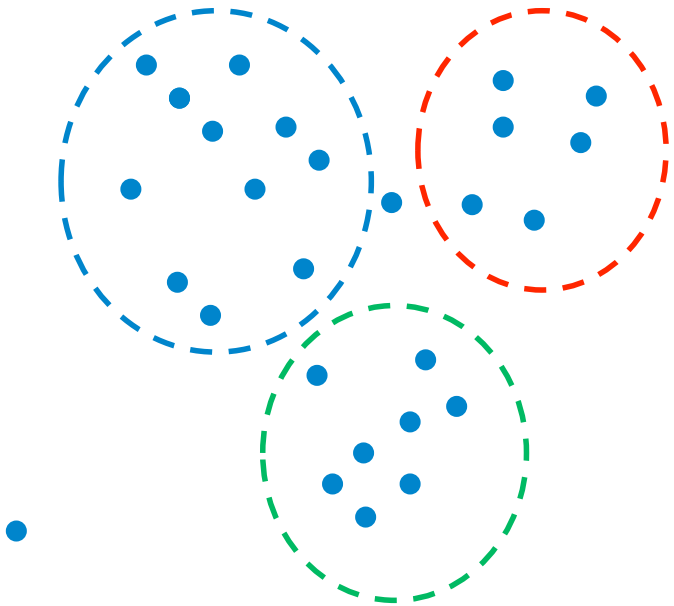
Chiang Mai University
July-August 2017

Classification problem



- A population is assumed to be partitioned in c groups or classes
- Let $\Omega = \{\omega_1, \ldots, \omega_c\}$ denote the set of classes
- Each instance is described by
  - A feature vector $\boldsymbol{x} \in \mathbb{R}^p$
  - A class label $y \in \Omega$
- Problem: given a learning set $\mathcal{L} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, predict the class label of a new instance described by $\boldsymbol{x}$

Clustering problem



- n objects described by
  - Attribute vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ (attribute data) or
  - Dissimilarities (proximity data)
- Goal: find a meaningful structure in the data set, usually a partition into c subsets, or a more complex mathematical representation (fuzzy partition, etc.)

Why can belief functions be useful?

1. Exploit the high expressiveness of belief functions to
   (a) Represent more faithfully the uncertainty of the predictions made by a classifier (for, e.g., combining several classifiers, or providing the user with richer information about the uncertainty of the classification)
   (b) Reveal richer information about the data (clustering problems)
2. Represent uncertainty about the data themselves:
   (a) Uncertain class labels (partially supervised learning)
   (b) Clustering of imprecise/uncertain data

Main approaches to classification

1. Classifier fusion: convert the outputs from standard classifiers into belief functions and combine them using, e.g., Dempster's rule (e.g., Quost al., 2011)
2. Develop evidential classifiers directly providing belief functions as outputs:
   (a) Distance-based classifiers: evidential K-NN rule (Denœux, 1995), evidential neural network classifier (Denœux, 2000)
   (b) Predictive evidential classifiers (e.g., logistic regression, Xu et al., 2015)

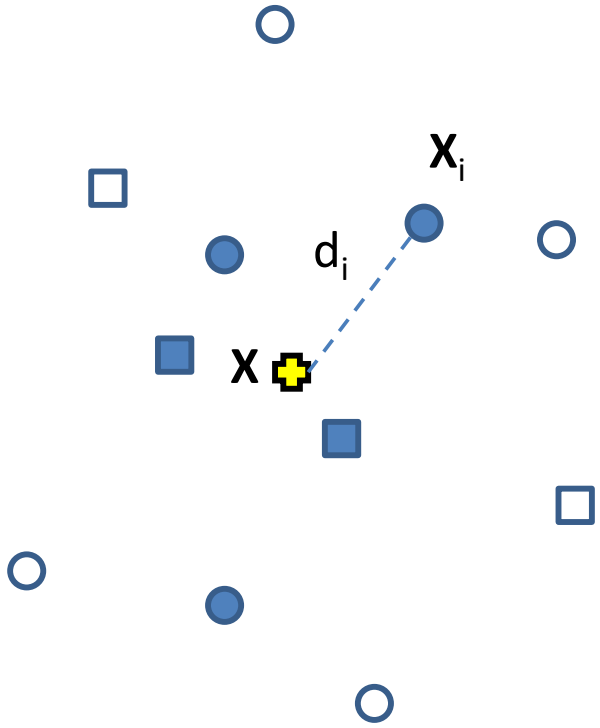
Outline

1. Evidential classification
   - Evidential K-NN rule
   - Evidential neural network classifier

Outline

Outline

Principle



- Let $\mathcal{N}_K(\boldsymbol{x}) \subset \mathcal{L}$ denote the set of the K nearest neighbors of $\boldsymbol{x}$ in $\mathcal{L}$, based on some distance measure
- Each $\boldsymbol{x}_i \in \mathcal{N}_K(\boldsymbol{x})$ can be considered as a piece of evidence regarding the class of $\boldsymbol{x}$
- The strength of this evidence decreases with the distance d_i between $\boldsymbol{x}$ and $\boldsymbol{x}_i$

Definition

- If $y_i = \omega_k$, the evidence of $(\boldsymbol{x}_i, y_i)$ can be represented by

$$\begin{aligned} m_i(\{\omega_k\}) &= \varphi_k(d_i) \\ m_i(\{\omega_\ell\}) &= 0, \quad \forall \ell \neq k \\ m_i(\Omega) &= 1 - \varphi(d_i) \end{aligned}$$

  where φ_k, $k = 1, \ldots, c$ are decreasing functions from $[0, +\infty)$ to $[0, 1]$ such that $\lim_{d \to +\infty} \varphi_k(d) = 0$

- The evidence of the K nearest neighbors of $\boldsymbol{x}$ is pooled using Dempster's rule of combination

$$m = \bigoplus_{\boldsymbol{x}_i \in \mathcal{N}_K(\boldsymbol{x})} m_i$$

- Choice of functions φ_k: for instance, $\varphi_k(d) = \alpha \exp(-\gamma_k d^2)$.
- Parameters $\gamma_1, \ldots, \gamma_c$ can be optimized (see below).

Learning

- Parameter $\gamma = (\gamma_1, \ldots, \gamma_c)$ can be learnt from the data by minimizing the following cost function

$$C(\gamma) = \sum_{i=1}^{n} \sum_{k=1}^{c} (pl_{(-i)}(\omega_k) - t_{ik})^2,$$

  where
  - $pl_{(-i)}$ is the contour function obtained by classifying $\boldsymbol{x}_i$ using its K nearest neighbors in the learning set.
  - $t_{ik} = 1$ is $y_i = k$, $t_{ik} = 0$ otherwise.
- Function $C(\gamma)$ can be minimized by an iterative nonlinear optimization algorithm.

Computation of $pl_{(-i)}$

- Contour function from each neighbor $\boldsymbol{x}_j \in \mathcal{N}_K(\boldsymbol{x}_i)$:

$$pl_j(\omega_k) = \begin{cases} 1 & \text{if } y_j = \omega_k \\ 1 - \varphi_k(d_{ij}) & \text{otherwise} \end{cases}, \quad k = 1, \ldots, c$$

- Contour function of the combined mass function

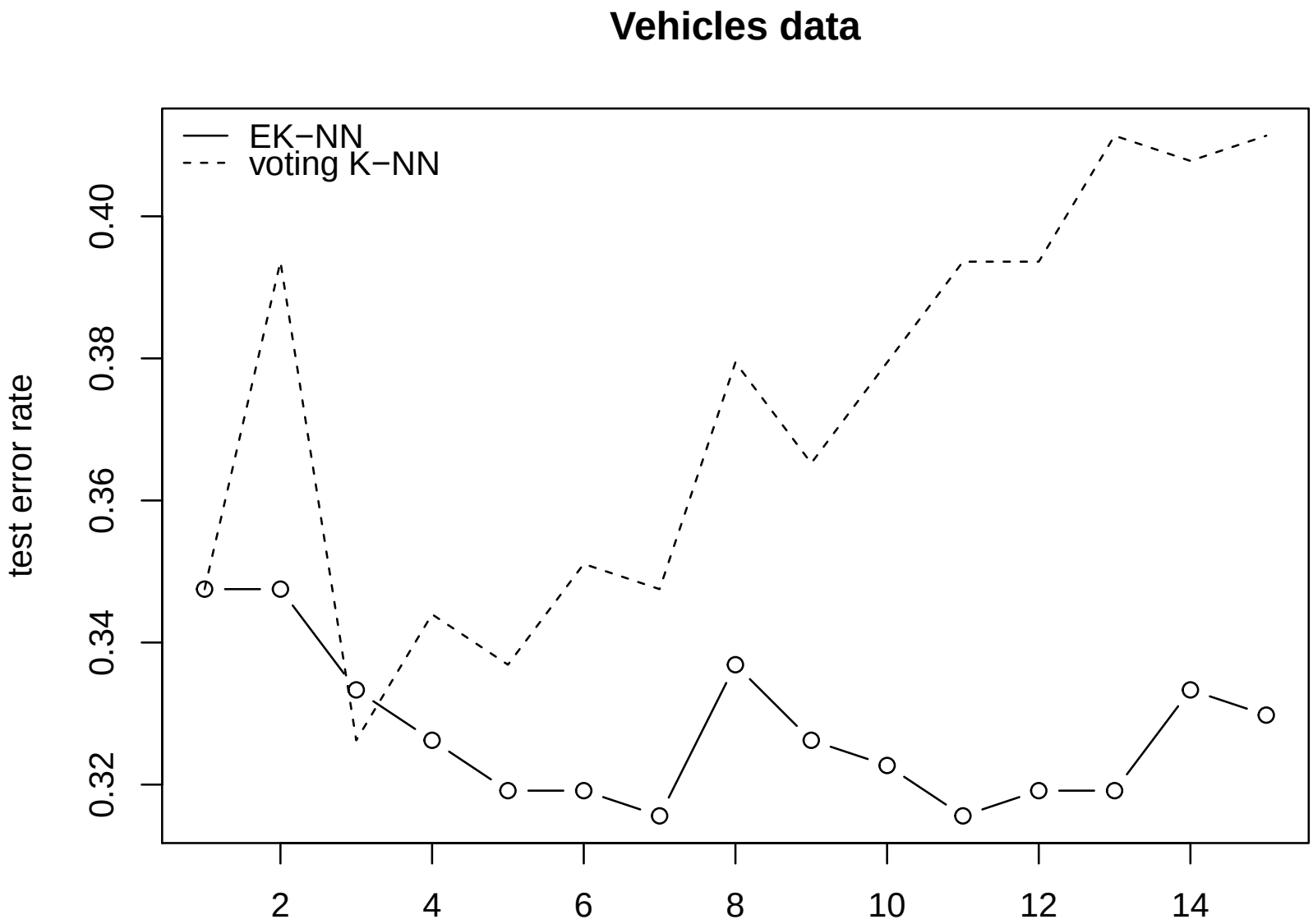
$$pl_{(-i)}(\omega_k) \propto \prod_{\boldsymbol{x}_j \in \mathcal{N}_K(\boldsymbol{x}_i)} (1 - \varphi_k(d_{ij}))^{1 - t_{jk}}$$

  where $t_{jk} = 1$ if $y_j = \omega_k$ and $t_{jk} = 0$ otherwise
- It can be computed in time proportional to $K|\Omega|$

Example 1: Vehicles dataset

- The data were used to distinguish 3D objects within a 2-D silhouette of the objects.
- Four classes: bus, Chevrolet van, Saab 9000 and Opel Manta.
- 846 instances, 18 numeric attributes.
- The first 564 objects are training data, the rest are test data.

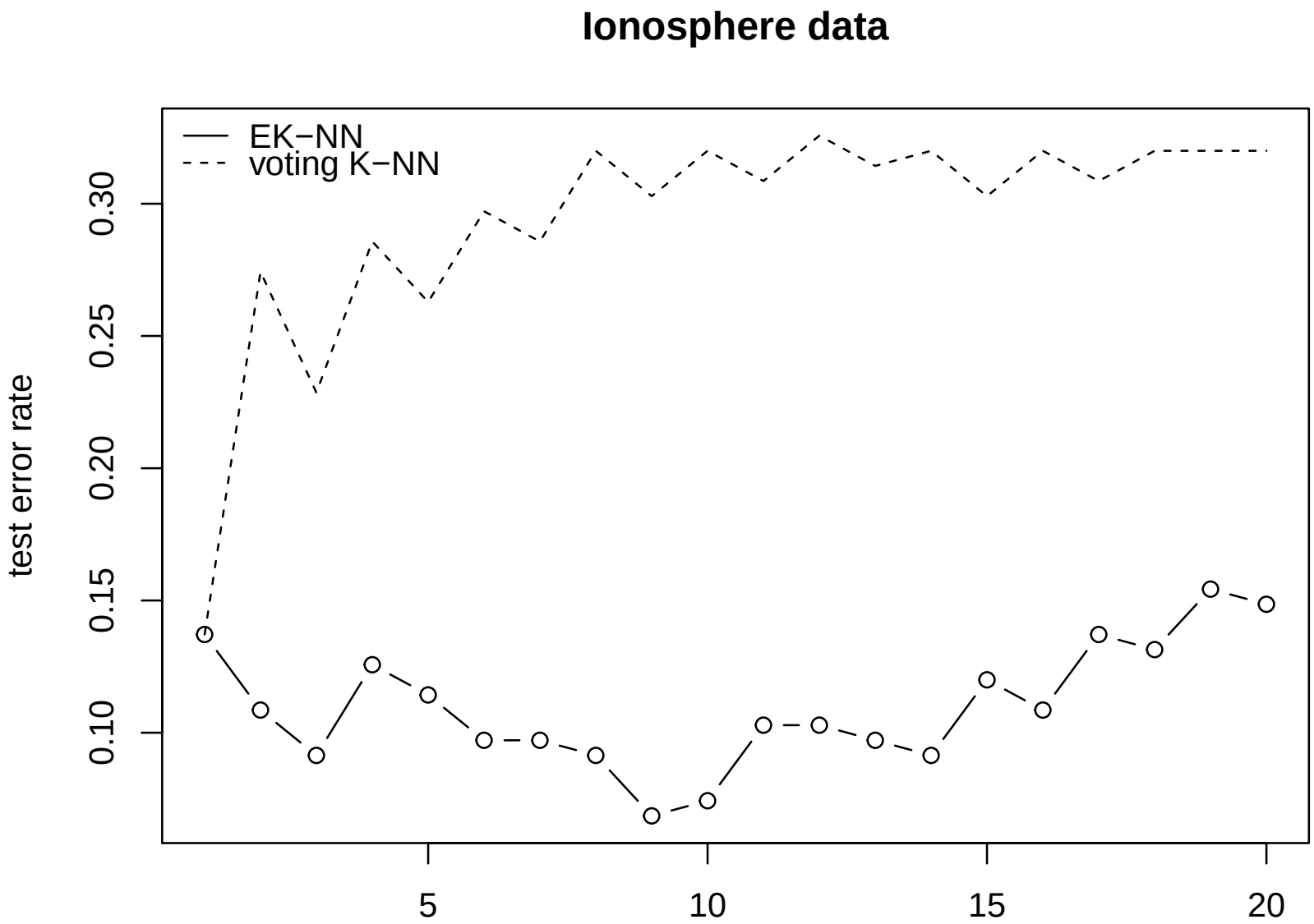
Vehicles datasets: result



Example 2: Ionosphere dataset

- This dataset was collected by a radar system and consists of phased array of 16 high-frequency antennas with a total transmitted power of the order of 6.4 kilowatts.
- The targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not.
- There are 351 instances and 34 numeric attributes. The first 175 instances are training data, the rest are test data.

Ionosphere datasets: result



Implementation in R

```
library("evclass")

data("ionosphere")
xapp<-ionosphere$x[1:176,]
yapp<-ionosphere$y[1:176]
xtst<-ionosphere$x[177:351,]
ytst<-ionosphere$y[177:351]

opt<-EkNNfit(xapp,yapp,K=10)
class<-EkNNval(xapp,yapp,xtst,K=10,ytst,opt$param)

> class$err
0.07428571
> table(ytst,class$ypred)
ytst  1  2
1   106  6
2   7  56
```

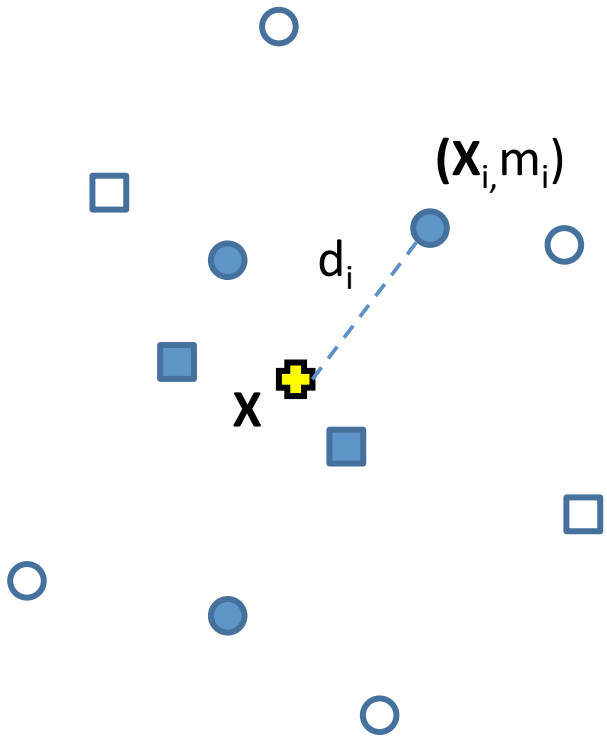
Partially supervised data

- We now consider a learning set of the form

$$\mathcal{L} = \{(\boldsymbol{x}_i, m_i), i = 1, \ldots, n\}$$

  where
  - $\boldsymbol{x}_i$ is the attribute vector for instance i, and
  - m_i is a mass function representing uncertain expert knowledge about the class y_i of instance i
- Special cases:
  - $m_i(\{\omega_k\}) = 1$ for all i: supervised learning
  - $m_i(\Omega) = 1$ for all i: unsupervised learning

Evidential k-NN rule for partially supervised data



- Each mass function m_i is discounted (weakened) with a rate depending on the distance d_i

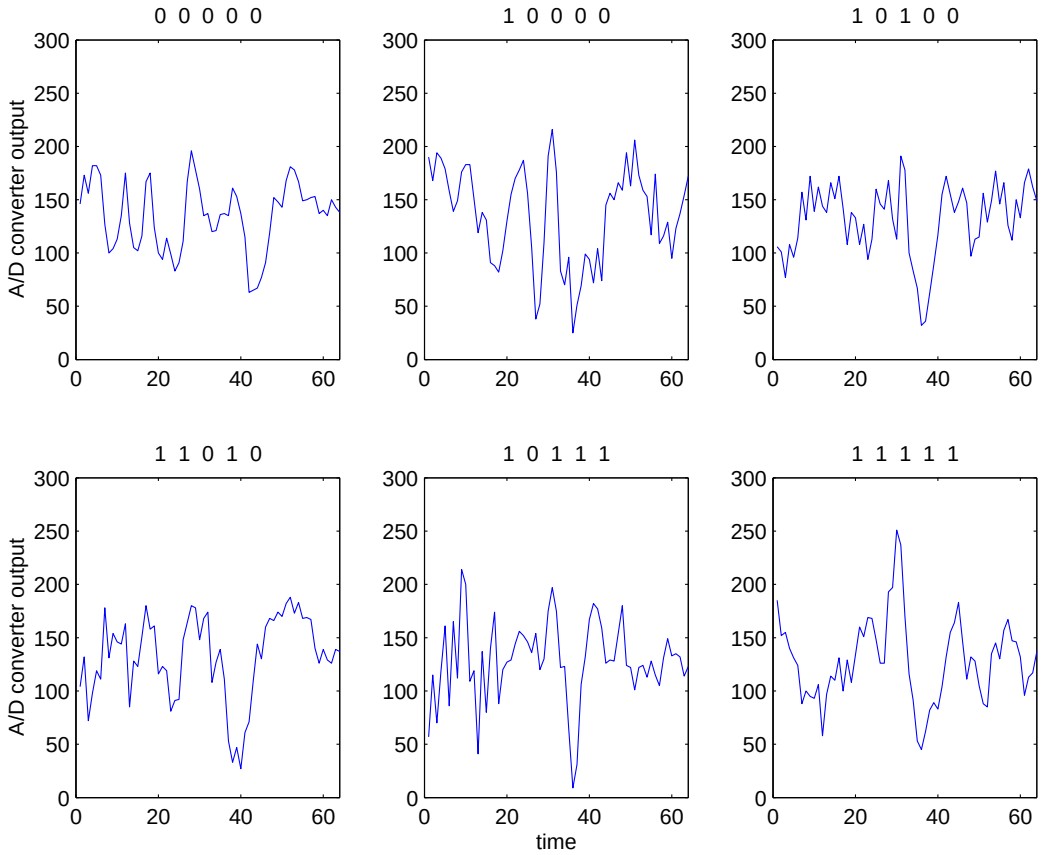
$$m'_i(A) = \varphi(d_i)\, m_i(A), \quad \forall A \subset \Omega$$
$$m'_i(\Omega) = 1 - \sum_{A \subset \Omega} m'_i(A)$$

- The K mass functions m'_i are combined using Dempster's rule

$$m = \bigoplus_{\boldsymbol{x}_i \in \mathcal{N}_K(\boldsymbol{x})} m'_i$$

Example: EEG data

EEG signals encoded as 64-D patterns, 50 % positive (K-complexes), 50 % negative (delta waves), 5 experts.



Results on EEG data

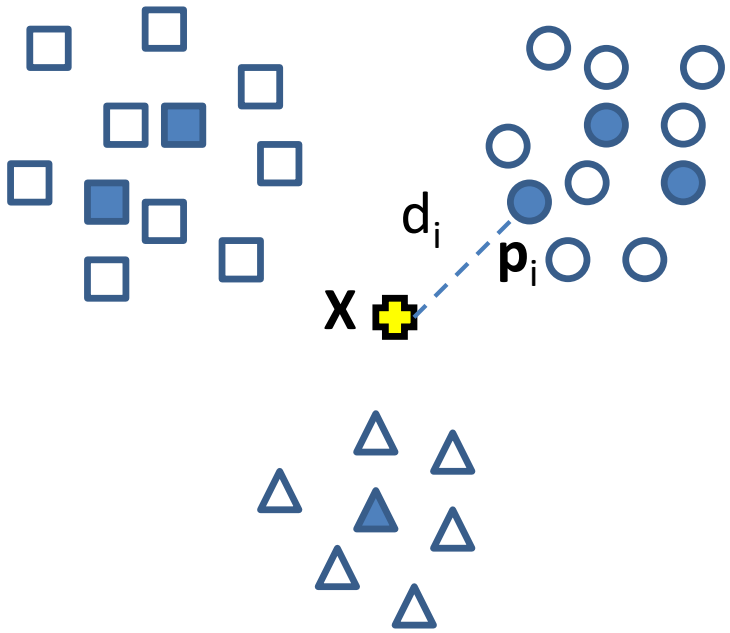
(Denoeux and Zouhal, 2001)

- $c = 2$ classes, $p = 64$
- For each learning instance $\boldsymbol{x}_i$, the expert opinions were modeled as a mass function m_i.
- $n = 200$ learning patterns, 300 test patterns

K	K-NN	w K-NN	Ev. K-NN (crisp labels)	Ev. K-NN (uncert. labels)
9	0.30	0.30	0.31	0.27
11	0.29	0.30	0.29	0.26
13	0.31	0.30	0.31	0.26

Outline

Principle



- The learning set is summarized by r prototypes.
- Each prototype $\boldsymbol{p}_i$ has membership degree u_{ik} to each class ω_k, with $\sum_{k=1}^{c} u_{ik} = 1$.
- Each prototype $\boldsymbol{p}_i$ is a piece of evidence about the class of $\boldsymbol{x}$, whose reliability decreases with the distance d_i between $\boldsymbol{x}$ and $\boldsymbol{p}_i$.

Propagation equations

- Mass function induced by prototype $\boldsymbol{p}_i$:

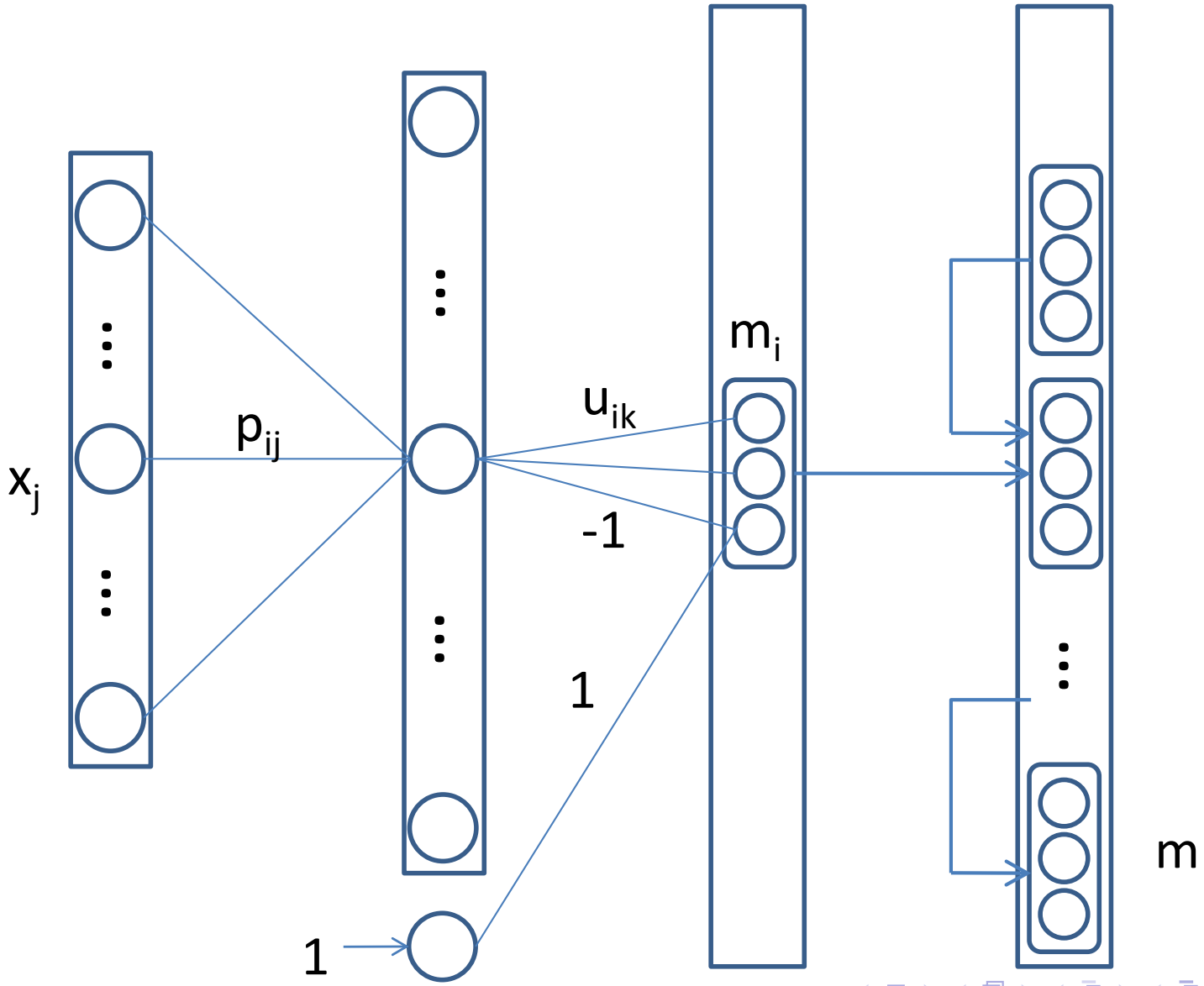
$$m_i(\{\omega_k\}) = \alpha_i u_{ik} \exp(-\gamma_i d_i^2), \quad k = 1, \ldots, c$$
$$m_i(\Omega) = 1 - \alpha_i \exp(-\gamma_i d_i^2)$$

- Combination:

$$m = \bigoplus_{i=1}^{r} m_i$$

- The computation of m_i requires $O(rp)$ arithmetic operations (where p denotes the number of inputs), and the combination can be performed in $O(rc)$ operations. Hence, the overall complexity is $O(r(p+c))$ operations to compute the output for one input pattern.
- The combined mass function m has as focal sets the singletons $\{\omega_k\}$, $k = 1, \ldots, c$ and Ω.

Neural network implementation



Learning

- The parameters are the
  - The prototypes $\boldsymbol{p}_i$, $i = 1, \ldots, r$ (rp parameters)
  - The membership degrees u_{ik}, $i = 1, \ldots, r$, $k = 1 \ldots, c$ (rc parameters)
  - The α_i and γ_i, $i = 1 \ldots, r$ ($2r$ parameters).
- Let θ denote the vector of all parameters. It can be estimated by minimizing a cost function such as

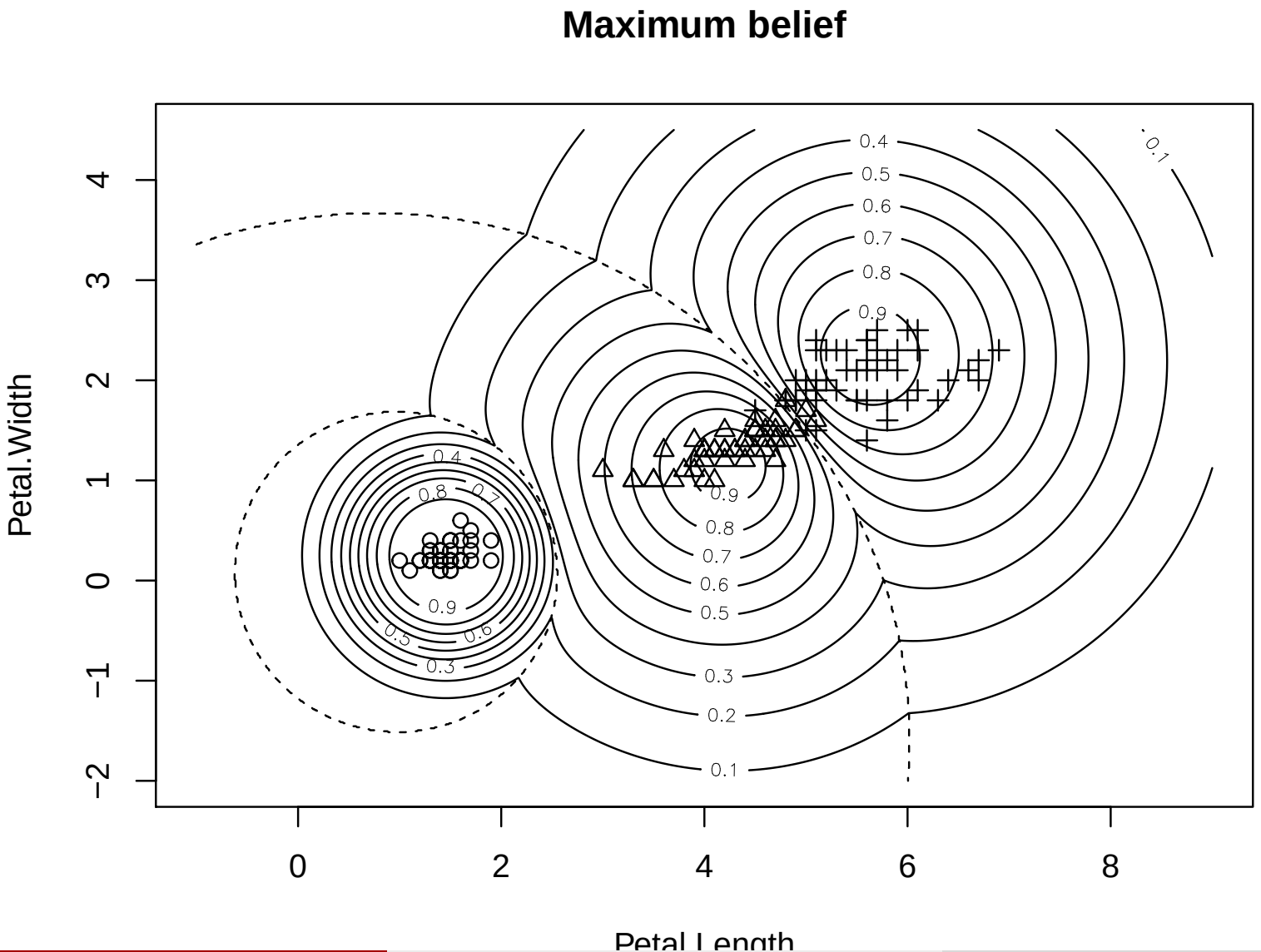
$$C(\theta) = \sum_{i=1}^{n} (pl_{ik} - t_{ik})^2 + \mu \sum_{i=1}^{r} \alpha_i$$

  where pl_{ik} is the output plausibility for instance i and class k, $t_{ik} = 1$ if $y_i = k$ and $t_{ik} = 0$ otherwise, and μ is a regularization coefficient (hyperparameter).
- The hyperparameter μ can be optimized by cross-validation.

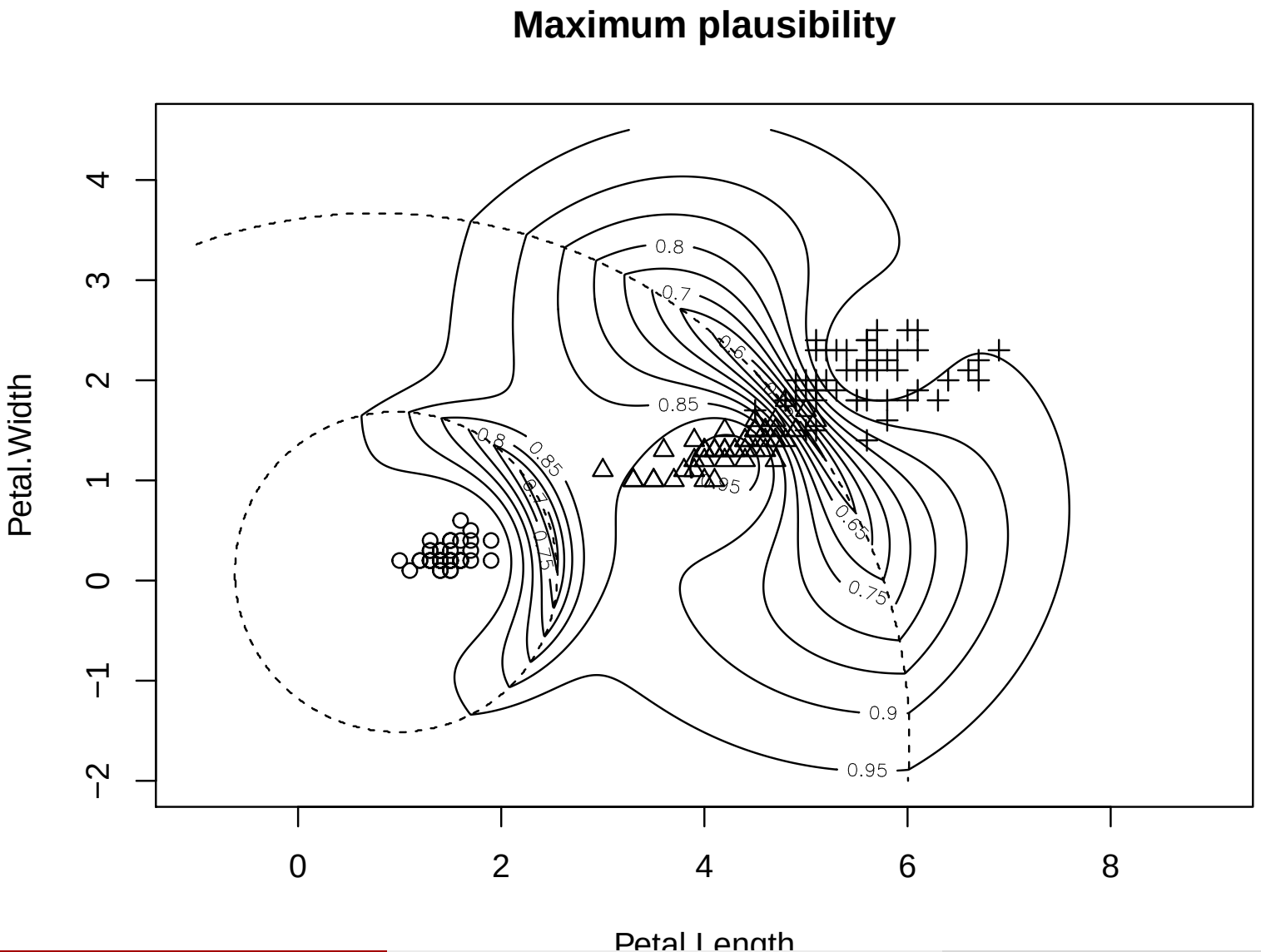
Implementation in R

```
library("evclass")

data(glass)
xtr<-glass$x[1:89,]
ytr<-glass$y[1:89]
xtst<-glass$x[90:185,]
ytst<-glass$y[90:185]

param0<-proDSinit(xtr,ytr,nproto=7)
fit<-proDSfit(x=xtr,y=ytr,param=param0)
val<-proDSval(xtst,fit$param,ytst)

> print(val$err)
0.3333333 > table(ytst,val$ypred)
ytst 1 2 3 4
1 30 6 4 0
2 6 27 1 3
3 4 3 1 0
4 0 5 0 6
```

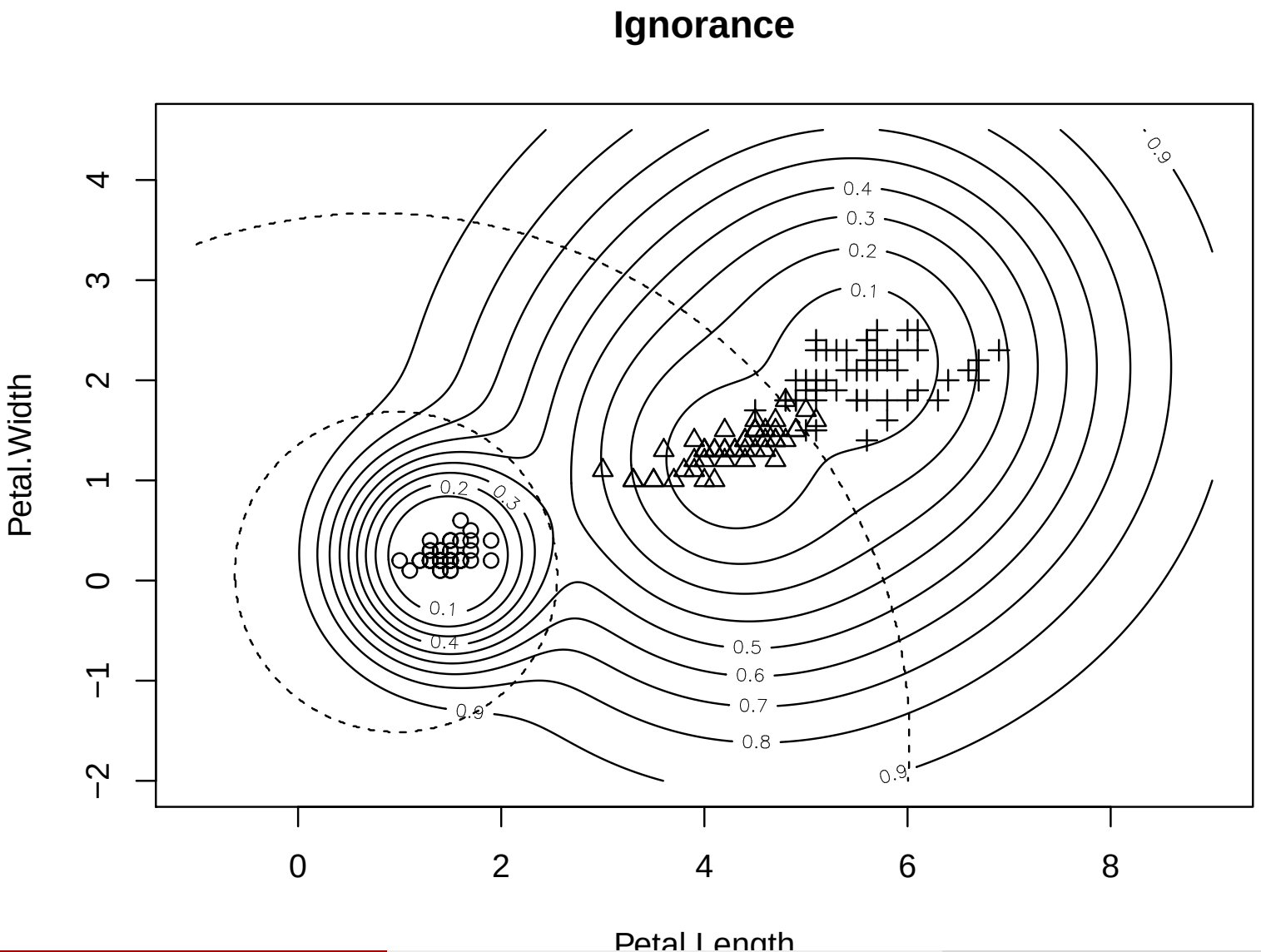
Results on the Iris data



Results on the Iris data



Results on the Iris data

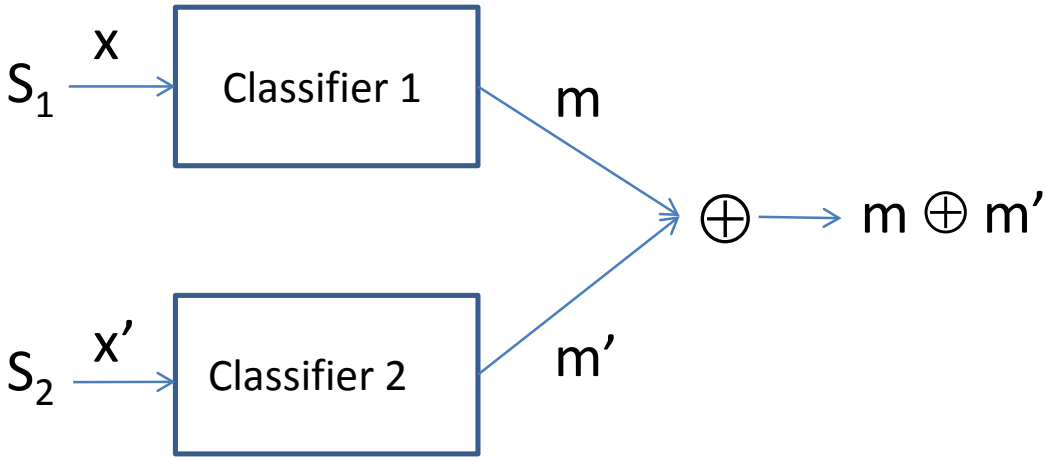


Results on classical data

Vowel data
$c = 11$,
$p = 10$
$n = 568$
test: 462 ex.
(different
speakers)

Classifier	test error rate
Multi-layer perceptron (88 units)	0.49
Radial Basis Function (528 units)	0.47
Gaussian node network (528 units)	0.45
Nearest neighbor	0.44
Linear Discriminant Analysis	0.56
Quadratic Discriminant Analysis	0.53
CART	0.56
BRUTO	0.44
MARS (degree=2)	0.42
Evidential NN (33 prototypes)	0.38
Evidential NN (44 prototypes)	0.37
Evidential NN (55 prototypes)	0.37

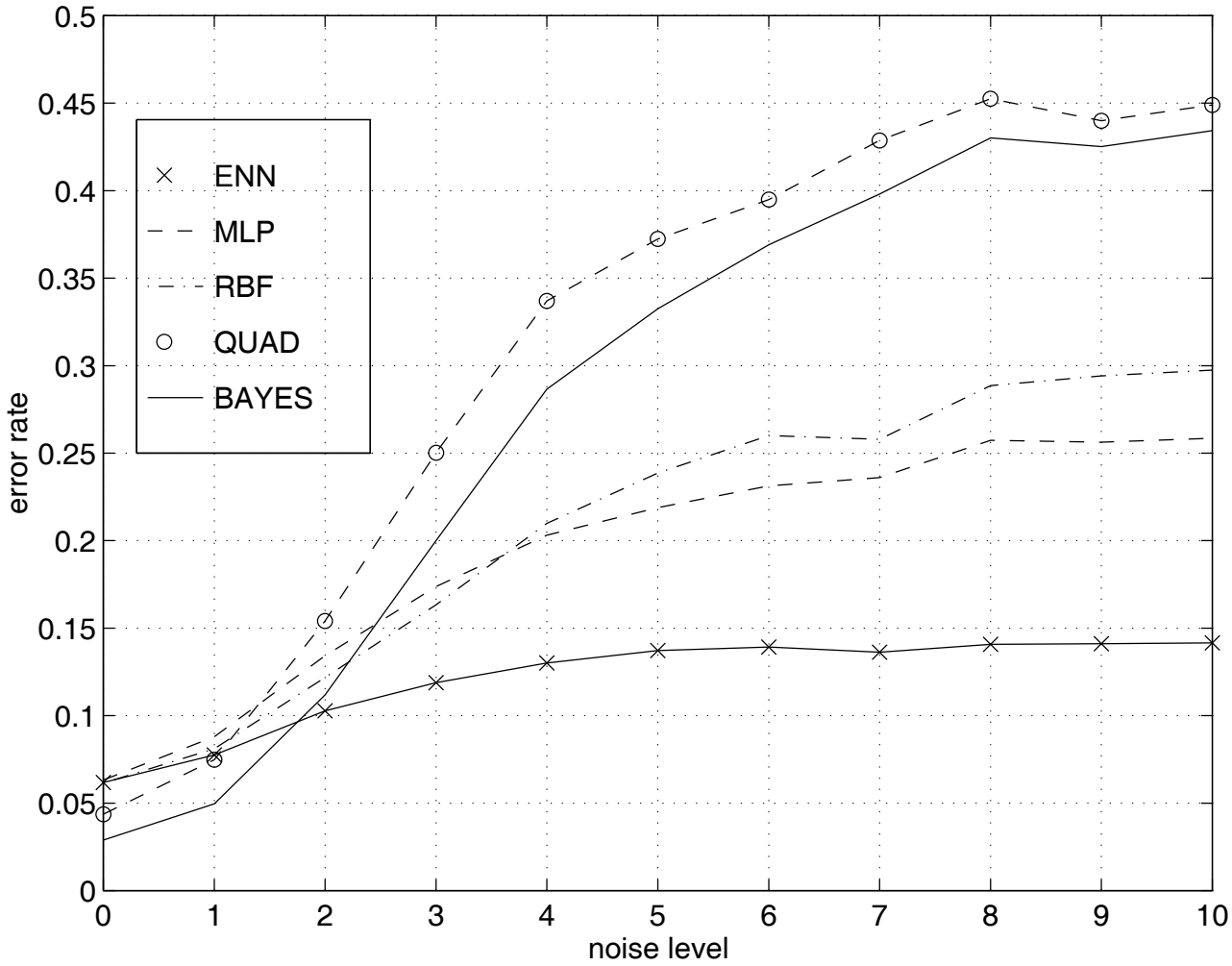
Data fusion example



- $c = 2$ classes
- Learning set ($n = 60$): $\boldsymbol{x} \in \mathbb{R}^5$, $\boldsymbol{x}' \in \mathbb{R}^3$, Gaussian distributions, conditionally independent
- Test set (real operating conditions): $\boldsymbol{x} \leftarrow \boldsymbol{x} + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$

Results

Test error rates: $\boldsymbol{x} + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$



References I

cf. https://www.hds.utc.fr/~tdenoeux

T. Denœux.

A k-nearest neighbor classification rule based on Dempster-Shafer theory.

IEEE Transactions on SMC, 25(05):804-813, 1995.

T. Denœux.

A neural network classifier based on Dempster-Shafer theory.

IEEE transactions on SMC A, 30(2):131-150, 2000.