Design of robust pattern classifiers based on optimum-path forests

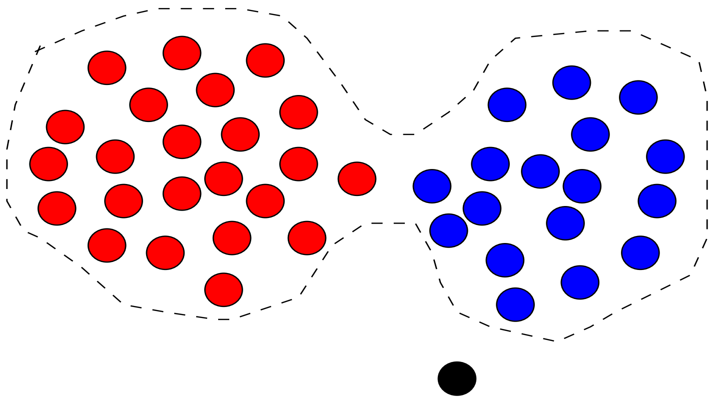
João P. Papa[1], Alexandre X. Falcão[1], Paulo A. V. Miranda[1], Celso T. N. Suzuki[1]
Nelson D. A. Mascarenhas[2]

[1]State University of Campinas, Institute of Computing, Campinas, Brazil
[2]Federal University of São Carlos, Department of Computing, São Carlos, Brazil

Presentation Overview

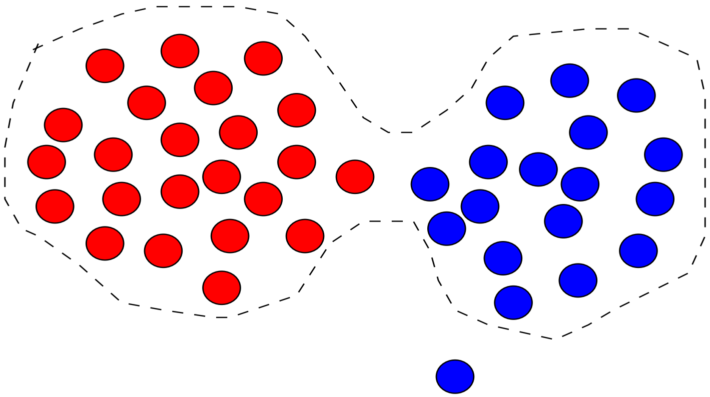
- Introduction
- OPF
- Experimental Results
- Conclusion and future work

Introduction



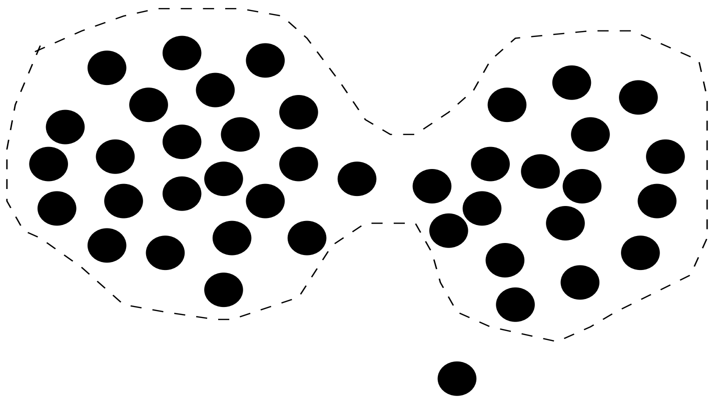
- Problem (Z_1 and Z_2)

Introduction



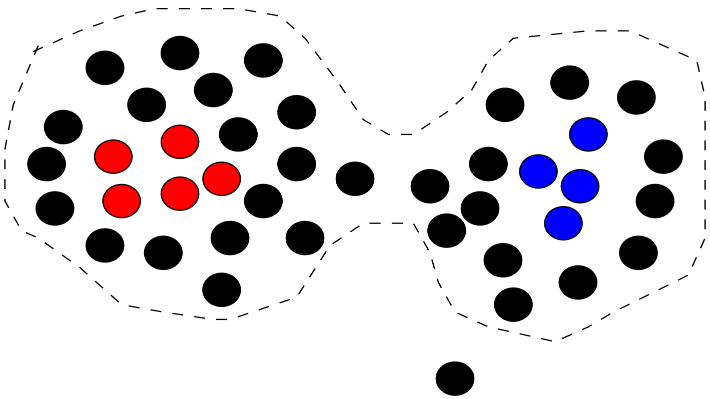
- Problem (Z_1 and Z_2)
- Pattern classification

Introduction



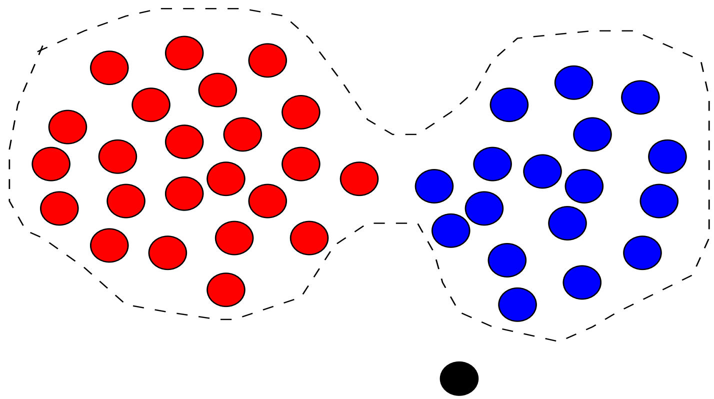
- Problem (Z_1 and Z_2)
- Pattern classification
- Unsupervised classification

Introduction



- Problem (Z_1 and Z_2)
- Pattern classification
- Unsupervised classification
- Semi-supervised classification

Introduction



- Problem (Z_1 and Z_2)
- Pattern classification
- Unsupervised classification
- Semi-supervised classification
- Supervised classification

Motivation

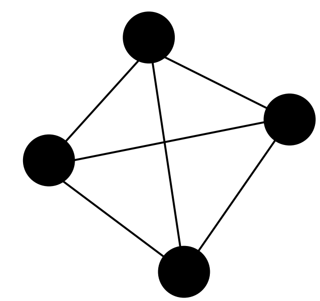
- To propose a new supervised classifier based on optimum path forest
- Support Vector Machines (SVM)
  - binary classifier
  - high dimensional space
- Artificial Neural Networks with Multilayer Perceptron (ANN-MLP)
  - unstable classifier
  - slow convergence

Optimum Path Classifier - OPF

- Watershed computed by the Image Foresting Transform (IFT) with markers obtained from Z_1 (training set) in the feature space

Modeling the problem

- samples are the nodes of the graph
- adjacency relation: complete graph
- arc weight $w(s,t) = d(\vec{s},\vec{t})$
- path-cost function f_{max}
- prototypes (markers) set S.



Optimum Path Forest - OPF

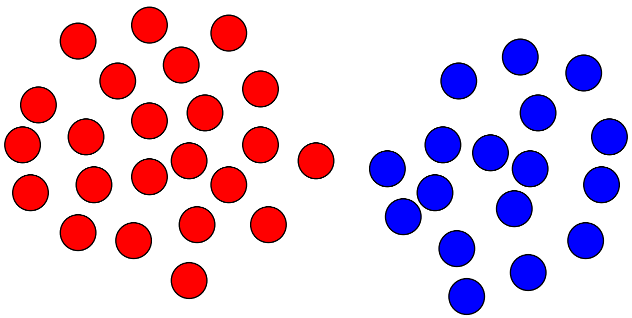
Supervised pattern classifier with 2 phases:

- Training: forest computation
- Unseen test: nodes are added to the forest, classified and removed

Main question in the training phase: how to choose the prototypes set S?

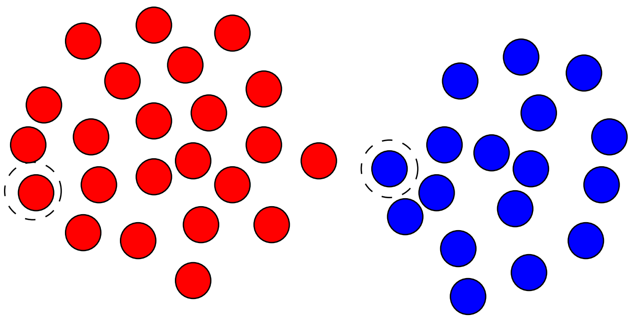
- random choice
- density choice
- minimum spanning tree (MST) choice

Training phase



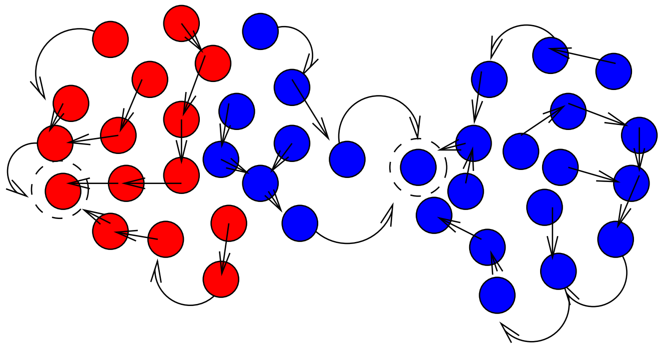
- Samples

Training phase



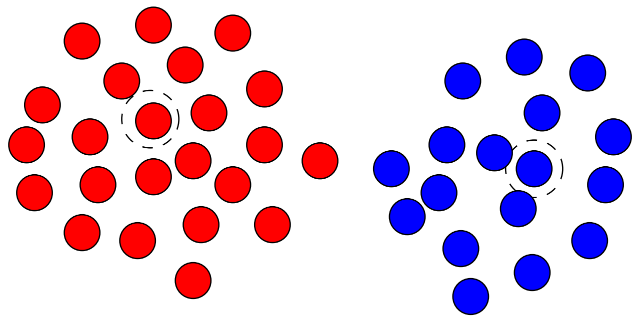
- Samples
- Random choice

Training phase



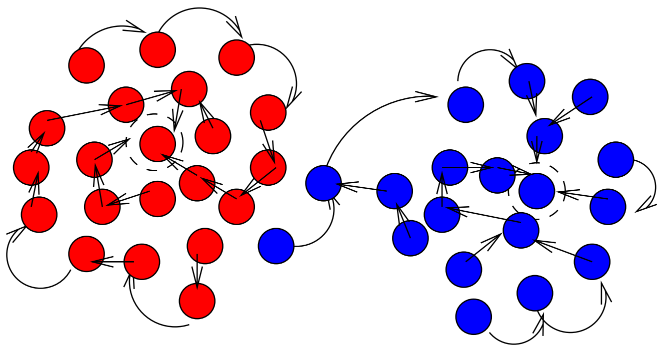
- Samples
- Random choice
- Random choice result

Training phase



- Samples
- Random choice
- Random choice result
- Density choice

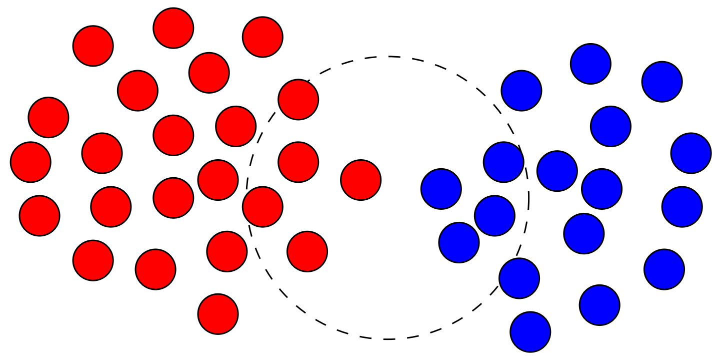
Training phase



- Samples
- Random choice
- Random choice result
- Density choice
- Density choice result

Training phase

Goal: to achieve zero error in the training set. How ??



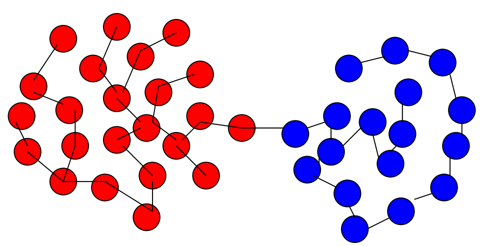
Problem region

To put prototypes inside the problem region! How can we identify them?

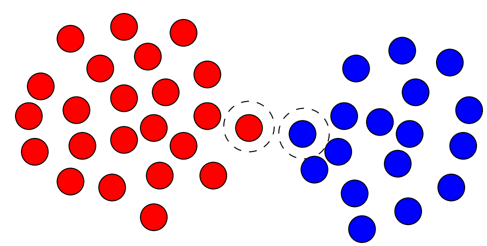
Training phase

MST approach

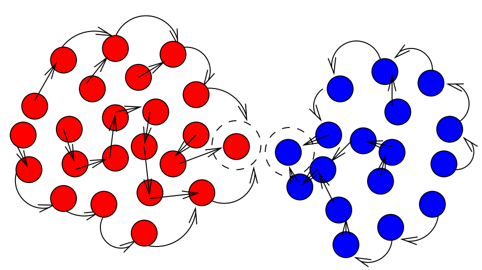
- sum of the weights of the edges is minimum
- each pair of nodes is connected by an optimum path



(a) MST



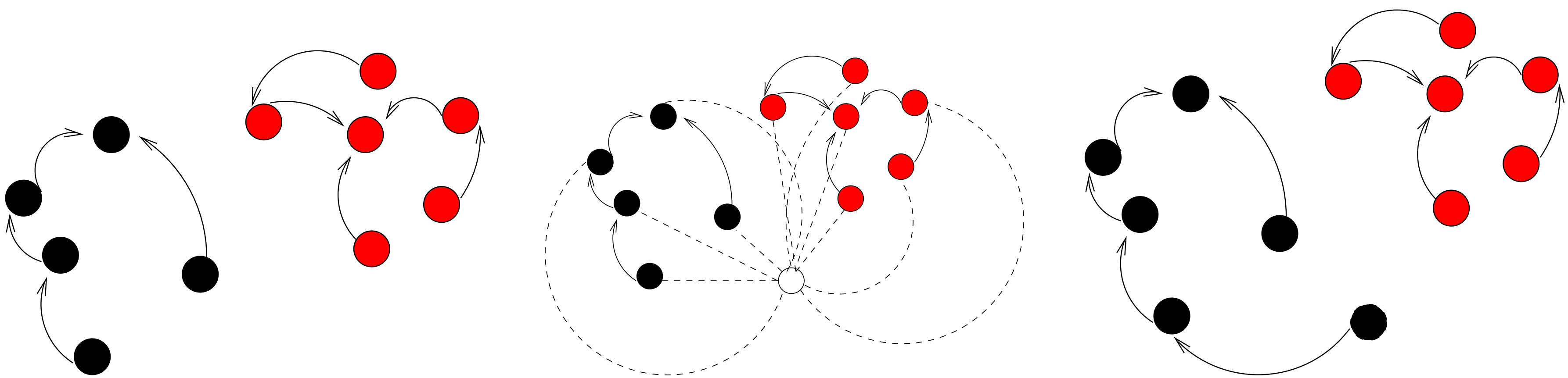
(b)Prototypes chosen by the MST



OPF nodes classification result

Test phase

- unseen samples are tested individually



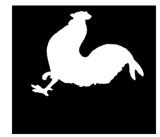
(a)Optimum path forest (b)Test sample (c)Classification result.

Experimental Results

We performed tests in 16 databases:

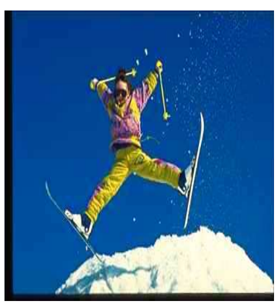
- MPEG-7: shape database containing 1400 objects equally distributed in 70 classes.





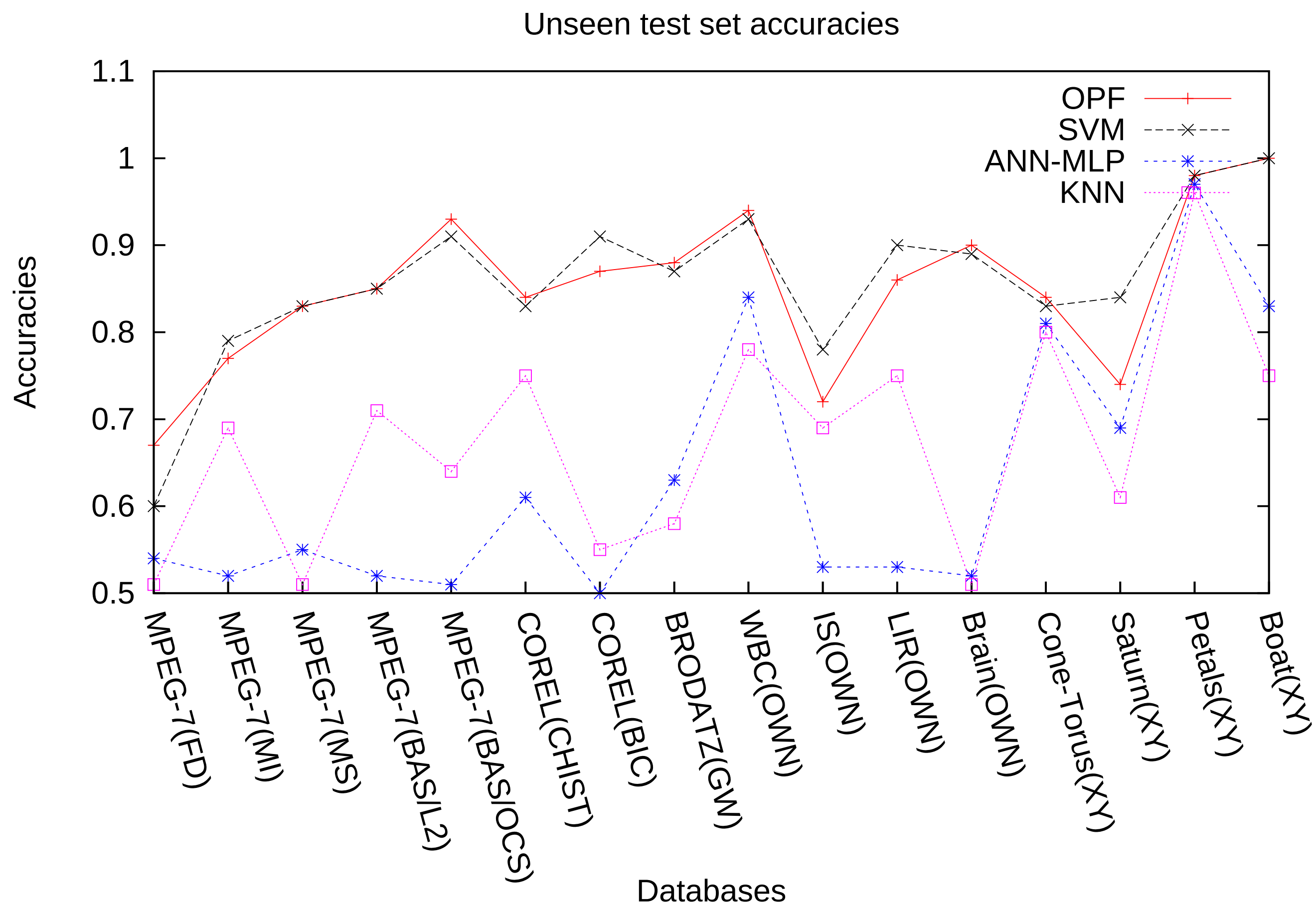

Fish 1 Fish 2 Chicken 1 Chicken 2

- Corel: database containing 1607 images of several objects distributed in 49 classes.




Ski 1 Ski 2 Pumpkin 1 Pumpkin 2

Experimental Results



OPF: 9 wins, 1 tie and 6 loses

Learning approach

How can we make sure that a classifier can learns with its own errors without increasing the training set size?

- Z_2: evaluation set

Learning approach

How can we make sure that a classifier can learns with its own errors without increasing the training set size?

- Z_2: evaluation set
- Learning algorithm: to identify more informative samples

Learning approach

How can we make sure that a classifier can learns with its own errors without increasing the training set size?

- Z_2: evaluation set
- Learning algorithm: to identify more informative samples
- Replacements between samples and errors

Learning approach

How can we make sure that a classifier can learns with its own errors without increasing the training set size?

- Z_2: evaluation set
- Learning algorithm: to identify more informative samples
- Replacements between samples and errors
- OPF is designed in Z_1 (training set) and Z_2 (evaluation set) and tested in the unseen Z_3 (test set)

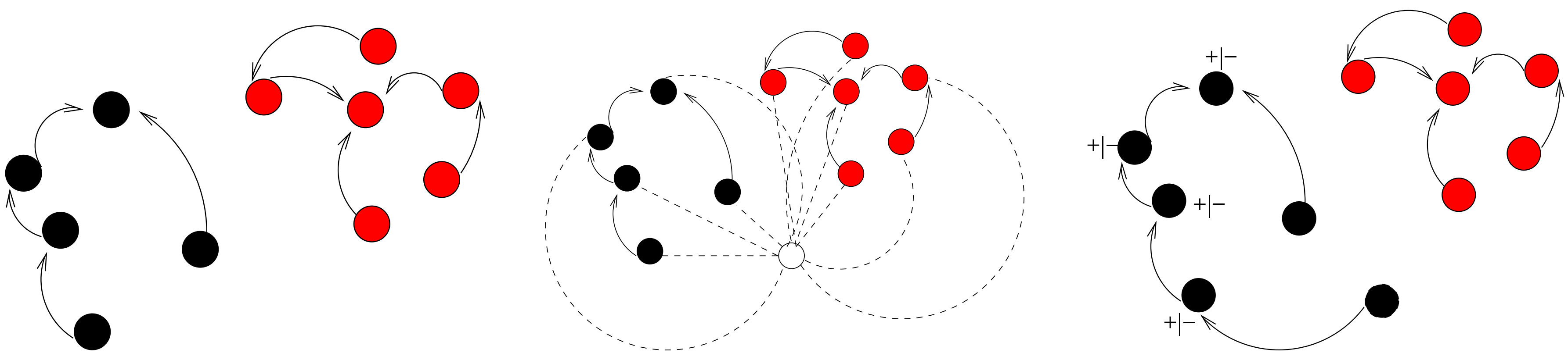
Learning approach

How can we make sure that a classifier can learns with its own errors without increasing the training set size?

- Z_2: evaluation set
- Learning algorithm: to identify more informative samples
- Replacements between samples and errors
- OPF is designed in Z_1 (training set) and Z_2 (evaluation set) and tested in the unseen Z_3 (test set)

Test phase

- unseen samples are tested individually
- relevance number
- irrelevant nodes



(a)Optimum path forest (b)Test sample (c)Reward/Penalty.

Learning algorithm

Algorithm:

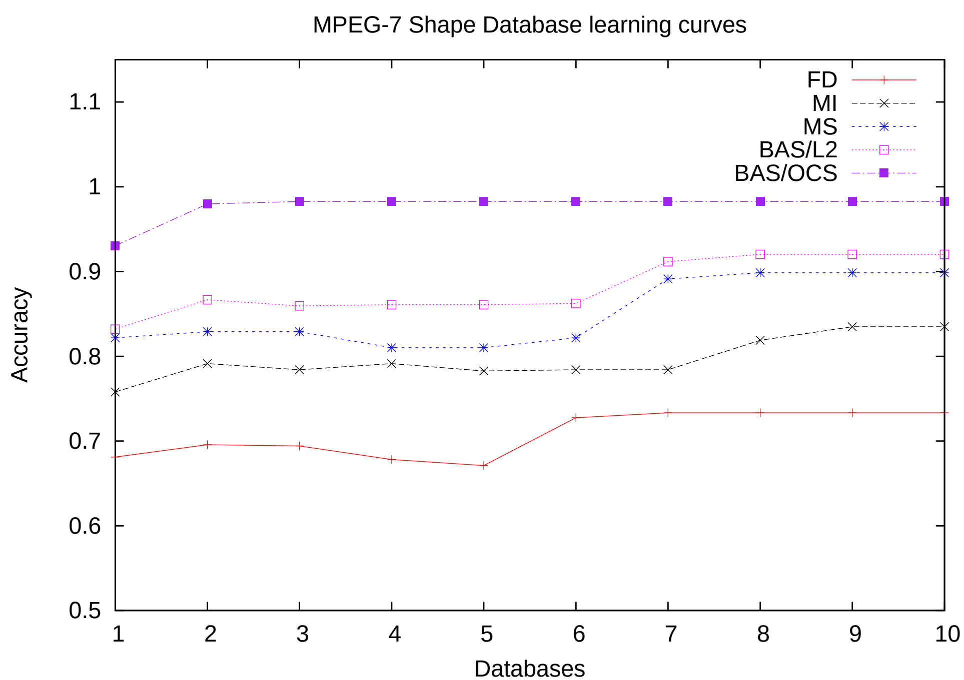
1.For I from 1 to N do

2. Build the classifier using the OPF algorithm (MST in Z_1).

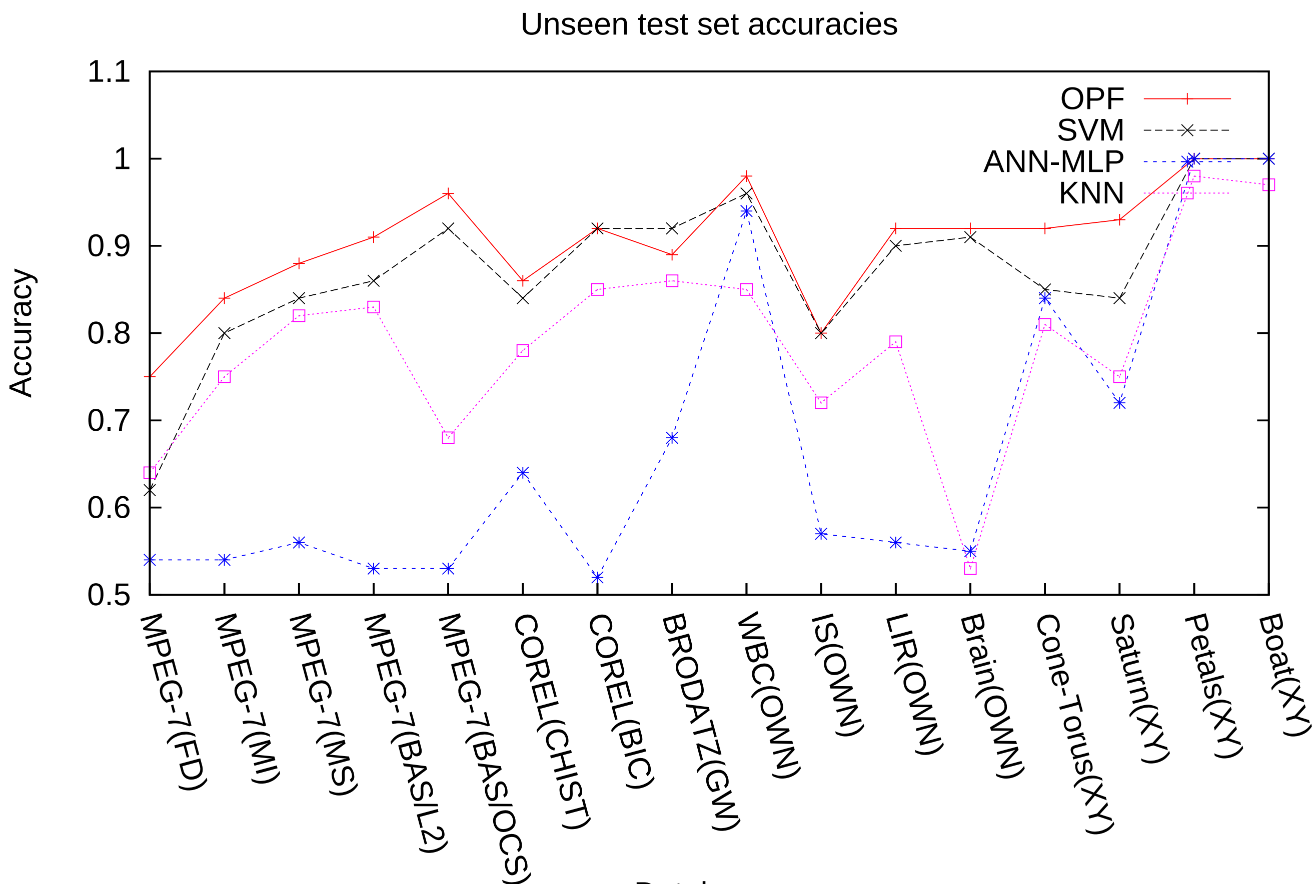
3. Classify samples in Z_2 and compute the relevance number for each sample in Z_1.

4. Replace misclassified elements in Z_2 by irrelevant (not prototypes) in Z_1.

5. If there exists irrelevant elements in Z_1, replace them by random samples from Z_2.

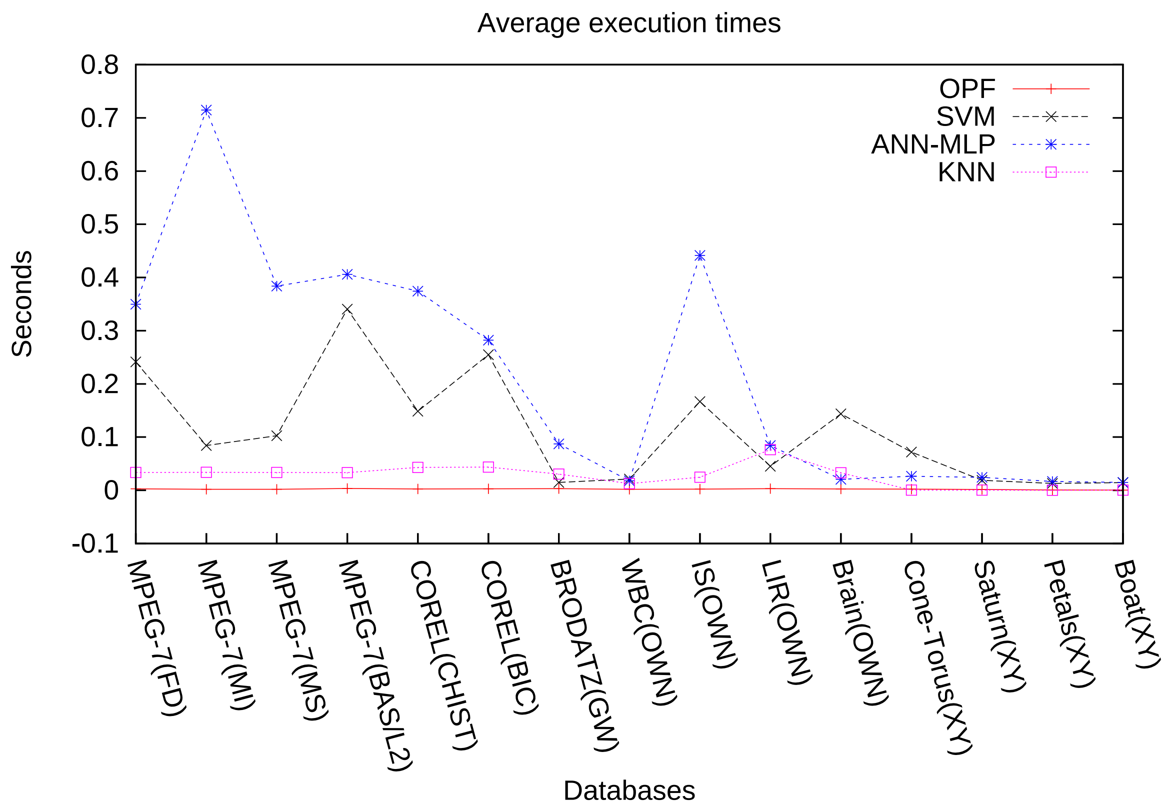


Experimental Results



OPF: 11 wins, 4 ties and 1 lose

Execution Times



The OPF was 47.21 times faster than SVM, 98.71 times faster than ANN-MLP and 7.81 times faster than KNN.

Conclusion and future works

- OPF is a new promising tool for supervised pattern recognition
- Faster than the tested approaches
- Similar to SVM (at least)
- Descriptor combination by genetic programming
- New path-cost functions