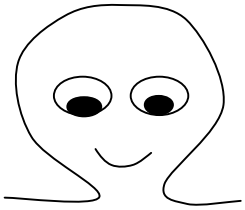
Meta-learning: searching in the model space.

Włodzisław Duch & Karol Grudziński,



Department of Informatics,
Nicholas Copernicus University,
Grudziądzka 5, 87-100 Toruń, Poland.

WWW: http://www.phys.uni.torun.pl/kmk

1. No free lunch theorem
2. Bias and the need for metalearning.
3. Framework for similarity-based methods
4. .. and what you can do with this.
5. Some results.
6. Some conclusions.

No free lunch theorem

Computational Intelligence should help with:

- classification: given vector X assign class C(X)
- heteroassociation: given vector X assign vector Y
- approximation: interpolate and extrapolate in N-D
- prediction: time series
- pattern completion: given $X=(X_n, X_u)$ find missing part X_u
- provide understandable explanation

Some Computational Intelligence methods:

- Neural networks – many types: MLP, RBF, recurrent...
- Fuzzy systems, rough systems.
- Neurofuzzy systems, adaptive fuzzy systems.
- Pattern recognition methods.
- Multivariate statistical methods: LDA, FDA, SVM ...
- Machine learning methods.
- Visualization methods.

Each method has its bias. Cf. StatLog comparison.

- No single method is the best for all data.
- For every method a data is found on which it works ..
- and a data on which it does not work!

What to do? Which method to choose?

- If we only could do it automatically ...

Alas! Only a few simple methods work automatically;
each method requires experience to use it.

Neural networks require too much experimentation – not suitable for automatic methods.
Decision trees? SVM, LDA, kernel methods?
Sometimes. Restricted application area.

Problems:
no common CI theory, most theories focused on learning;
different assumptions for different methods;
specialized programs for each task.

An expert system to advice on which method to use?
Difficult to create.

- Goal of metalearning:

automatically create the best method for a given problem.

- How to do it?

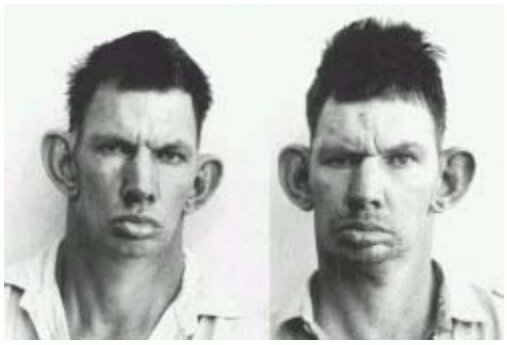
Try to unify many different methods in a single framework.
Search in the model space.

Good candidate for rich framework: similarity based methods.

Why similarity based approach?

Various names referring to the similarity based approach:

Memory-Based Methods (MBM),
Instance-Based Methods (IBM),
Case-Based Methods (CBM),
Case-Based Reasoning (CBR),
Memory-Based Reasoning (MBR),
Similarity-Based Reasoning (SBR).



Nearest Neighbor (NN) method - the simplest method of classification developed in pattern recognition.

Task:

Given a reference set of data vectors and classes

$$\{X^{(k)}, C_k = C(X^{(k)})\}$$

predict the class of a new vector X.

$d(X, X^{(k)})$ is similarity measure (distance function)

$p(C_i|X; M)$ is classification probability,
M = model parameters (function $d()$, procedures).

Similarity-based methods include all methods based on prototypes (kNN, RBF, LVQ, SOM, ...).

Methods using discriminant functions, such as LDA or MLP may also be based on similarity !

Example: kNN method

1. Pre-process all vectors: zero mean, standardize variance
2. Define similarity (distance) function

$$d(X,Y)=\sqrt{\sum_{i=1}^{N}(X_i-Y_i)^2}$$ Euclidean

$$d(X,Y)=\sum_{i=1}^{N}\left|X_i-Y_i\right|$$ Manhattan or City Block

$$d(X,Y)=\left(\sum_{i=1}^{N}\left(X_i-Y_i\right)^{\alpha}\right)^{1/\alpha}$$ Minkovsky

$$d(X,Y)=\max_{i=1..N}\left|X_i-Y_i\right|$$ Chebyschev, or $\alpha=\infty$

$$d(X,Y)=\sum_{i=1}^{N}\frac{\left|X_i-Y_i\right|}{\left|X_i+Y_i\right|}$$ Camberra

$$d(X,Y)=\sum_{j}\left[\sum_{i}\left|\Pr\left(C_j\middle|X_i\right)-\Pr\left(C_j\middle|Y_i\right)\right|\right]$$ Value Difference Metric

3. Perform classification:

For 1-NN

find $\min_k d\left(X,R^{(k)}\right)$, predict $C(X) = C(R^{(k)})$

$p(C_i \mid X; M) = 1$ for $C_i = C(R^{(k)})$

For k-NN method:

use k neighbors to determine the class;

select k to minimize the number of errors

if k_i vectors from class C_i have the same distance:

$p(C_i \mid X; M) = k_i / k$

• Advantages of kNN

Very simple, results should be used as a reference;
No learning phase (except to select k);
No parameters to manipulate;
Leave-one-out test is easily performed;
Usually good results, sometimes best of all classifiers
(if enough reference vectors provided);
Well-suited for industrial, large scale applications.

Many real-world applications

• Disadvantages

Large number of good examples is needed - for small datasets results may be poor;
No data compression, all reference vectors are kept;
Large computational demands in the classification phase,
Large memory requirements.
No automatic feature selection.

Decision regions: hypersurfaces, for 1-NN convex polygons

Asymptotic errors for $k \to \infty$ and for the number of reference vectors $\to \infty$ reach optimal Bayesian values.

In practice small k may work better.

A Framework for SB Methods

Prob. $p(C_i \mid X; M)$ of assigning class C_i to vector X

$$M = \left\{k, d(\cdot; r), G\left(d(\cdot)\right), \left\{R^n\right\}, p_i(R), E[\cdot], K(\cdot), \Re(\cdot), \{proc\}\right\}$$

$\boldsymbol{k}$	number of neighbors included,
$\boldsymbol{d(\cdot,r)}$	similarity function with parameter r
$\boldsymbol{G(d(\cdot,r))}$	function that weights the contribution of $\boldsymbol{R}$
$\{\boldsymbol{R}^{(n)}\}$	a set of reference vectors (prototypes)
$\{\boldsymbol{p_i}(\boldsymbol{R}^{(n)})\}$	class probability for reference vector
$\boldsymbol{E[\cdot]}$	cost function K
$\boldsymbol{K(\cdot)}$	kernel function, scaling the influence of the error on the cost function
$\Re(\ )$	risk function

proc, various procedures such as:

- Learning methods.
- Methods to optimize model complexity.
- Mixture of many models M_l.
- Methods of feature/reference selection.
- Various network realizations.

- Learning

gradient minimization of cost function;
global optimization minimization methods;
Bayesian learning, with regularization;
Mean Field Theory learning.

General form of a cost function with a risk matrix:

$$E\left(\{X\};\Re,M\right)=\sum_i\sum_X \Re\left(C_i,C(X)\right)H\left(p(C_i|X;M),\delta(C_i,C(X))\right)$$

H – usually a quadratic function or entropy measure.

Select the best model from the SBM framework,
combining parameters/procedures.

- start from the simplest model;
- check results of adding parameters/procedures;
- select the best extension;
- repeat until no significant improvement is found.

Optimize different type of parameters only if
significant improvements of results are obtained.

Start with the simplest models, such as kNN or LVQ

- Few examples of SBM framework methods

$$M = \left\{k, d(\cdot;r), G\left(d(\cdot)\right), \left\{R^n\right\}, p_i(R), E[\cdot], K(\cdot), \Re(\cdot), \{proc\}\right\}$$

k-NN

$\boldsymbol{G(d(\cdot,r))}$ = hard sphere with k+1 vectors inside
Training vectors = the reference set $\{\boldsymbol{R}^{(n)}\}$
$\{\boldsymbol{p_i}(\boldsymbol{R}^{(n)})=1\}$, $\boldsymbol{K(\cdot)=I}$
$\boldsymbol{E[\cdot;k]}$ = number of errors in the leave-one-out

Realization: Hamming neural network for binary inputs

r-NN

$\boldsymbol{G(d(\cdot,r))}$ = hard sphere of radius r: optimize *r*
$\boldsymbol{k}$ – variable, some vectors may be rejected
Hard limit of RBF

Realization: Restricted Coulomb Energy (RCE) classifier
a mixture of r_l-NN models after initial clusterization

Soft weighting k-NN and r-NN algorithms:
Instead of the hard sphere weighting function use continuous $\boldsymbol{G()}$.

The conical radial function (triangular membership function):

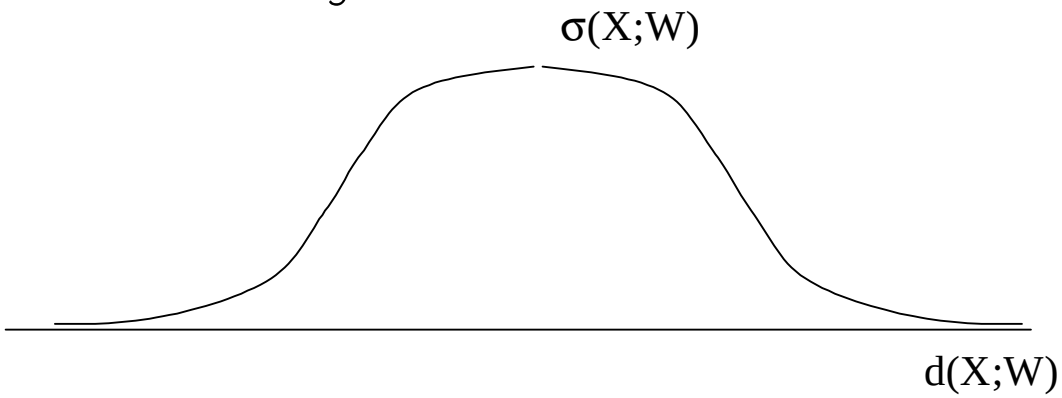
$$G(\mathbf{X},\mathbf{R};r) = \max\left(0,1 - d(\mathbf{X},\mathbf{R}) / r\right)$$

Gaussian, inverse multiquadratic or other radial f.

Bicentral function

$$G(\mathbf{X},\mathbf{R};r) = \sigma\left(\|\mathbf{X}-\mathbf{R}\| - r\right) - \sigma\left(\|\mathbf{X}-\mathbf{R}\| + r\right)$$

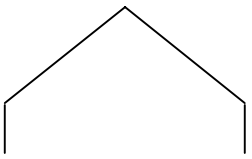
where σ is a sigmoidal function.



Many other localized functions.

In k-NN if r_k is the distance to k-th neighbor

$G(d;r_k,\alpha) = \max\left(0,1 - d / \alpha r_k\right)$, optimize α



For large α standard kNN (no weighting)
For α=1 last neighbor is not counted.

Parameterization of similarity measures

$$d(X,Y) = \sqrt{\sum_{i=1}^{N}(X_i - Y_i)^2}$$ Euclidean

$$d(X,Y) = \sum_{i=1}^{N}|X_i - Y_i|$$ Manhattan or City Block

$$d(X,Y) = \left(\sum_{i=1}^{N}(X_i - Y_i)^{\alpha}\right)^{1/\alpha}$$ Minkovsky

$$d(X,Y) = \sum_{j}\left[\sum_{i}\left|p(C_j \mid X_i) - p(C_j \mid Y_i)\right|\right]$$

Modified Value Difference Metric

Scaling different components of data vectors:

$$d(\mathbf{X},\mathbf{Y};\mathbf{s})^{\alpha} = \sum_{i=1}^{N} s_i |X_i - Y_i|^{\alpha}$$

Probabilistic scaled distance measures, like MVDM.

$$d(X,Y)^{\alpha} = \sum_{j}^{N}\sum_{i}^{K} s_i \left|p(C_i \mid X_j) - p(C_i \mid X_j)\right|^{\alpha}$$

for symbolic or discrete values.

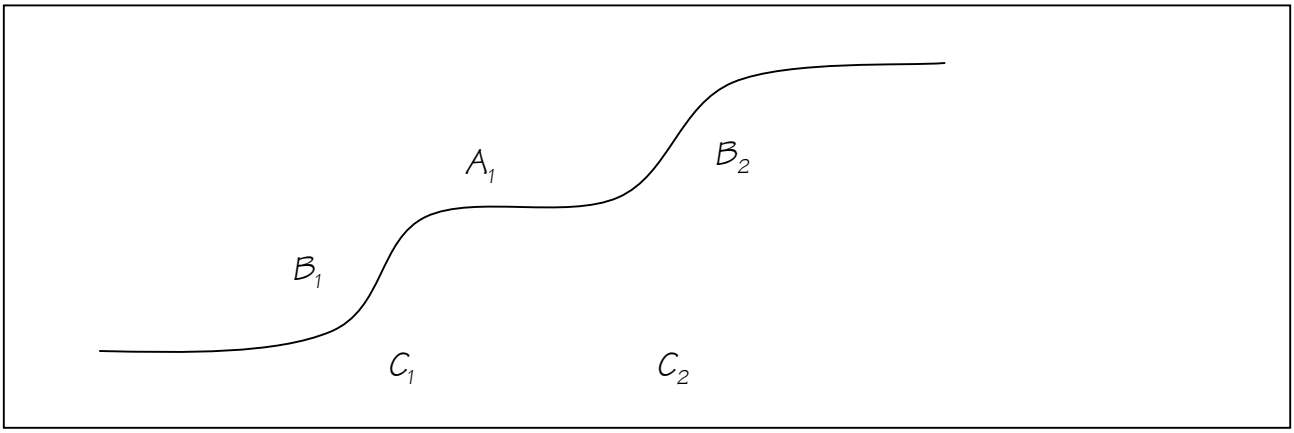
- More complex similarity functions.

MLP or any other adaptive continuous function may be used as a similarity function.

Train to minimize in-class distance variance and maximize between-class variance.

Soft multistep function – "neural" formulation. Combination of sigmoidal functions transforming features:

$$X_i \leftarrow \sum_{j=1}^{K_i} A_{ij}\sigma\left(B_{ij}X_i - C_{ij}\right)$$



- Active Selection of Reference Vectors

Select $\{\boldsymbol{R}^{(n)}\}$ from the training vectors:

add new vectors only if they improve results;
remove vectors that do not degrade results;
remove vectors with all k neighbors ∈ same class.

Focus on classification border: select nearest vectors from those that belong to other classes.

- Optimize reference vectors

$$\mathbf{R} \leftarrow \mathbf{R} + \eta\delta_{\pm}(C(\mathbf{X}), C(\mathbf{R}))(\mathbf{X} - \mathbf{R})$$

or use other LVQ techniques.

Add Virtual Support Vectors (VSV) at the border if necessary.

Repeat selection/optimization steps.

If LDA or SVM (single hyperplane) gives best results only 2 prototype vectors are sufficient.

P-rules (prototype-based rules)

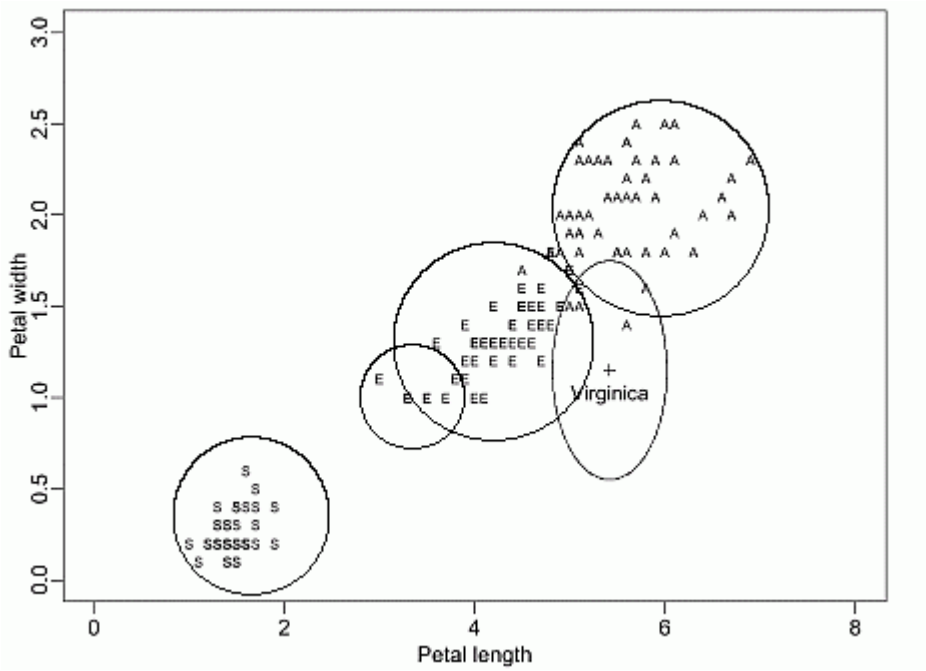
More general than fuzzy rules?
Distance functions => membership functions.

Several methods to create them:

- neural: RBF or LVQ-like, neurofuzzy
- decision trees approaches
- minimal distance or similarity-based approaches

- the nearest neighbor method, kNN, k=1
- optimization of distance function
- selection of relevant features.

Ex: prototypes for Iris—not at the centers of clusters!



SB methods and Neural Networks

Radial Basis Function neural networks (RBF) are a special case of MD method with

$G(d(\cdot,r))$ – Gaussian or other radial function,
d – Euclidean distance (other metric functions may be used).

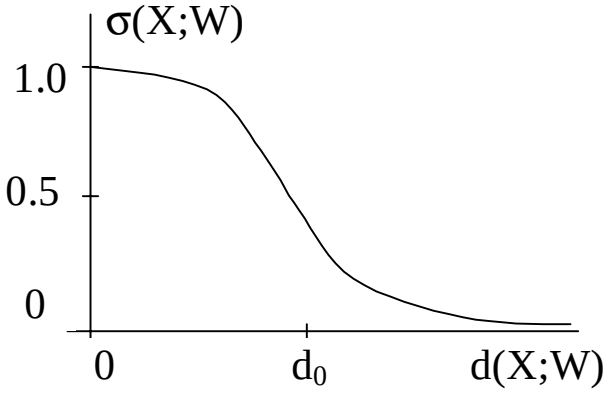
D-MLP, distance-based MLPs

$$\sigma(\mathbf{W}\cdot\mathbf{X}-\theta)=\sigma\left(\frac{1}{2}\left(\|\mathbf{W}\|^2+\|\mathbf{X}\|^2-\|\mathbf{W}-\mathbf{X}\|^2\right)-\theta\right)$$

$$G(D)=\sigma\left(d_0-D(\mathbf{W},\mathbf{X})\right)$$

$$d_0=\frac{1}{2}\left(\|W\|^2+\|X\|^2\right)-\theta$$

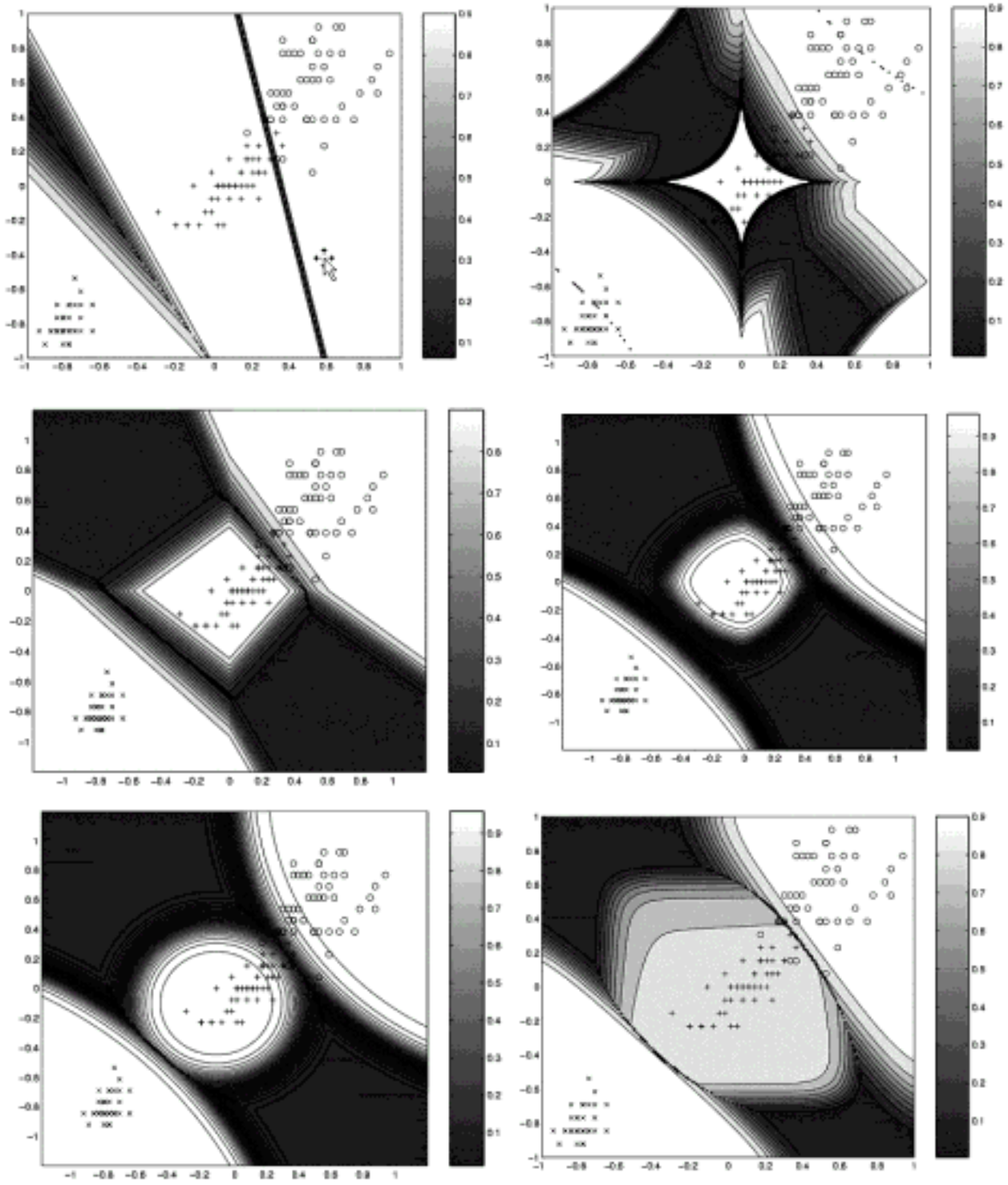
$$D(W,X)=\frac{1}{2}\left(\|W-X\|^2\right)$$



$d(\mathbf{X},\mathbf{Y})=\frac{1}{2}\sum_{i=1}^{N}|X_i-Y_i|^2$ corresponds to MLP with the standard scalar product activation

Sigmoidal functions scale the influence of references.
d_0 – scale factor, $\sigma(0)=0.5$ for $D(W, X) = d_0$

Minkovsky *distance*, MLP first, than α =0.5, 1, 1.5, 2, 7



Simplest SBM network realization

Hidden nodes compute distances D(X-R).
k nodes with min distances output class label $h_j(X;R)=C_i$
Others nodes $h_j(X;R)=\mathbf{0}$.

Output layer computes confidence factors:

$$P(C_i|\mathbf{X};M)=\sum_j W_{ij}\cdot h_j(\mathbf{X})$$

Output weights $W_{ij}=S(C_i,C_j)/C_j,\ C_i=1..N_C,$

$\mathbf{C_j}$ class vector in the r radius of X contributes $S(C_i,C_j)$,
where $S(\cdot,\cdot)$ = similarity among the output classes.

After normalization this network estimates probabilities:

$$p(C_i|\mathbf{X};M)=\frac{P(C_i|\mathbf{X};M)}{\sum_j P(C_j|\mathbf{X};M)}$$

Cost function used for training:

$$E(M(k,r,W))=\sum_i\sum_{\mathbf{X}} R(C_i,C(\mathbf{X}))\left(p(C_i|\mathbf{X},M)-\delta(C_i,C(\mathbf{X}))\right)^2$$

where $R(\cdot,\cdot)$ is the risk matrix.
Weights, k, r are treated as adaptive parameters.

Many other network realizations are possible.

Pattern completion and missing values.

In SBM partially known input $X=(X_d, X_u)$ may be used to find neighbors in the subspace of defined input X_d. Probability of unknown values X_u (multiple data imputation) is calculated by:

$$p(X_u|X_d; M) = \max_{u',i} p(C_i|(X_{u'}, X_d); M)$$

Echocardiogram data (UCI): 132 vectors, 12 attributes, only 1-9 useful, 2 classes.
15 values of A6 missing, 11 values A7 etc.

Hepatitis dataset (UCI): 155 vectors, 18 attributes, 13 binary, other integer, 2 classes.
A18 has 67 missing values, A16 has 29 etc.

10-fold stratified CV tests using FSM network

Method	Echocardiogram	Hepatitis
Ignored	87.8%, 24 nodes	79.9%, 19 nodes
Averages	85.5%, 20 nodes	81.0%, 12 nodes
New values	81.5%, 22 nodes	79.1%, 16 nodes
Maximize p	90.2%, 18 nodes	83.4% 10 nodes

Meta-learning algorithm.

Combine AI search and CI optimization approaches.

Given a framework for generation of CI models search the set of all possible models $\{M_a\}$:

start from simplest models,
add more procedures/parameters,
select the best models,
continue search until there is no improvement.

Best = simplest and most accurate, includes complexity. Avoid overfitting, check model complexity, use validation sets or criteria such as

Minimum Description Length (MDL)
Guaranteed Risk Minimization (GRM)
Information-based criteria (AIC, BIC, SIC).

CI models are homogenous, use the same type of parameters, add more elements to build decision borders.

Ex: Decision Trees create hyperboxes,
NN use soft hyperplanes,
RBF use Gaussians ...

Result: simple problems become complex.
Spherical distribution in N-dim. requires N+1 hyperplanes.
Single plane requires many Gaussians.

Evaluation function $C(M_l)$, ex. accuracy on validation set.
M initial models, select K final models.

The model sequence selection algorithm based on the Best-First Search (BFS) procedure is:

Create a pool of M initial models $\{M_l\}$, l=1 .. M.
Optimize and evaluate $C(M_l)$,
arrange models in a decreasing order $C_a(M_i) \geq C_a(M_j)$ for i>j.
Select best model M_R from the $\{M_l\}$ pool as the reference.
Remove it from the pool of models.

Repeat until the pool of models is empty:

Extend M_R:

$\forall M_l$ evaluate performance M_R+M_l
If there is a significant improvement:
  replace $M_R \leftarrow M_R+M_l$ with best extension, max $C(M_R+M_l)$
  remove M_l from the pool of available models.
ELSE stop, return M_R

M - L sequences evaluated per step, L = M-1, .. , 1.
Result: sequence of models of increasing complexity.

Warning: sequential optimization does not guarantee absolute optimum.

BFS – prone to local minima.
Beam search – more costly, but better results.
Select several models of similar accuracy.

Example:
Start from M_R = simplest k-nn, k=1, Euclidean.
$C(M_l)$ is the leave-1-out accuracy.

Model extensions include:

- Optimization of k, $k_1 \leq k \leq k_2$
- Selection of the type of $d(X,Y)$: Euclidean, Manhattan, Chebyschev and Camberra.
- Feature selection.
- Optimization of scaling factors.

$$d(\mathbf{X}, \mathbf{Y}; \mathbf{s})^{\alpha} = \sum_{i=1}^{N} s_i \left| X_i - Y_i \right|^{\alpha}$$

α = 1, 2 was used with scaling.

Parameter optimization:
simplex method + quantization of parameters, $s_i \leftarrow s_i \pm 0.1$
Sometimes gives rather large variance.

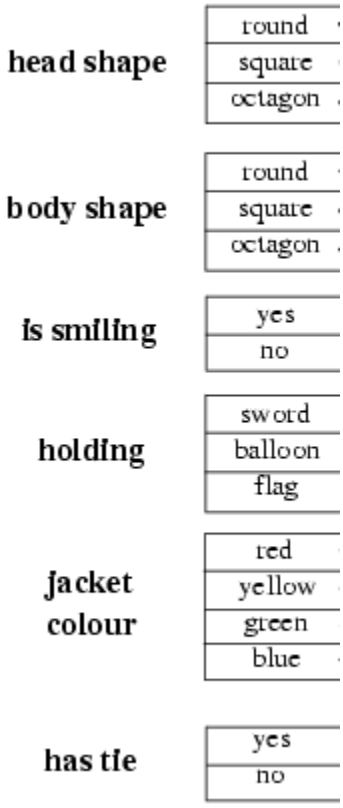
Monk problems

Artificial datasets used to test ML programs, data from UCI repository.

6 symbolic input features,
124 training cases,
432 test cases,
2 classes: monk, not-monk.

Monk-1 problem:
monk if (head shape = body shape) ∨
jacket color = red

Difficult for SBM, easy for rule-based.



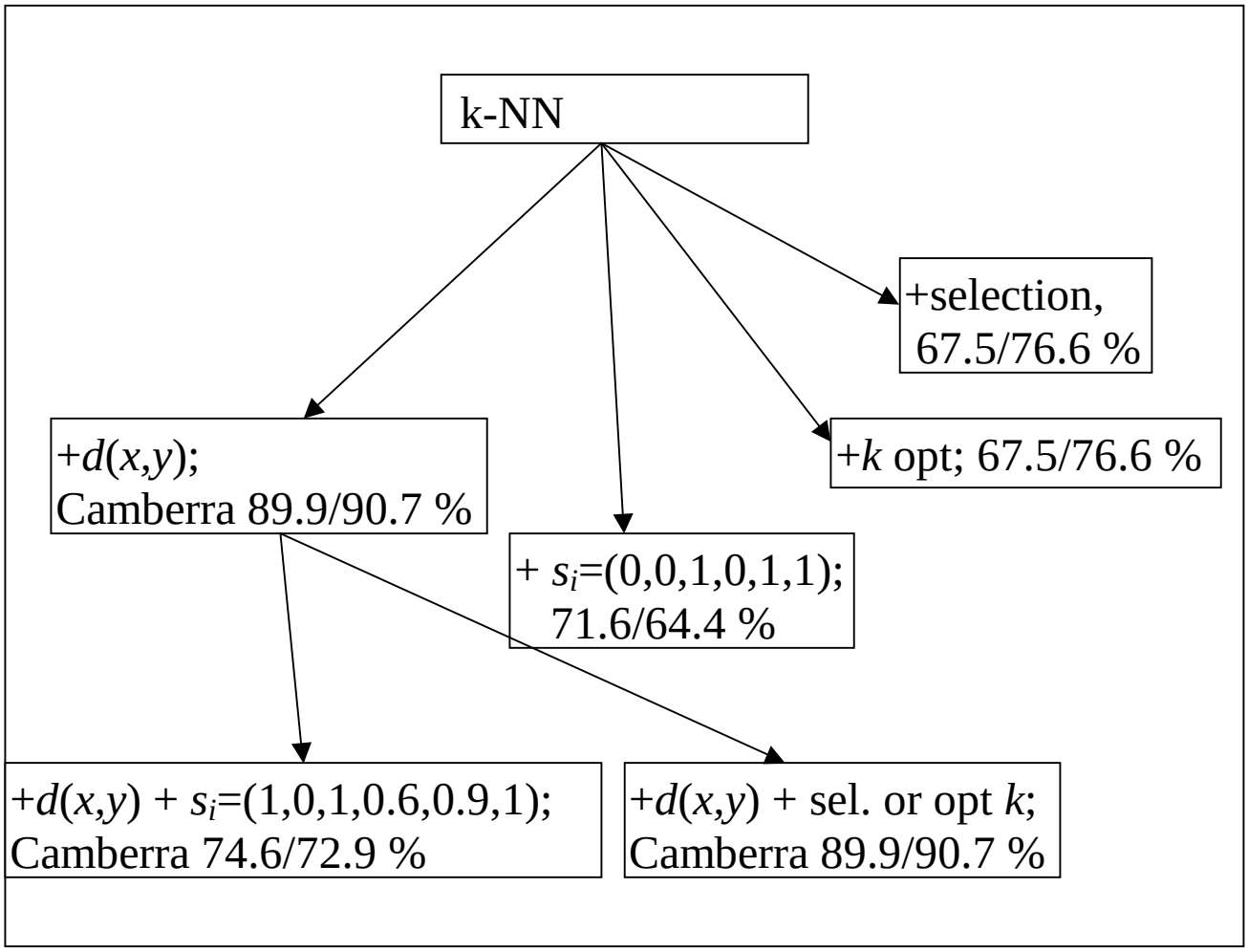
$C(M_l)$ is the leave-1-out accuracy.
Reference model: k-nn, k=1, Euclidean distance,
$C(M_l)$ on training 76.6% (test result 85.9%).

Method	$C(M_l)$ train %	Test acc. %
M_1 = k-nn, k=1, Euclidean	**76.6**	**85.9**
M_1+ Camberra distance	**79.8**	**88.4**
M_1+ k=3	**82.3**	**80.6**
M_1+ feature sel. 1, 2, 5	**96.8**	**100.0**
M_2=M_1+feature weights	**99.2**	**100.0**
new reference		
M_2+ Camberra distance	**100.0**	**100.0**

Weights selected: (1, 1, 0.1, 0, 0.9, 0)

Monk 2 problem:

Rule: Monk IF exactly 2 of 6 attributes have first value.
169 cases for training, 432 test cases



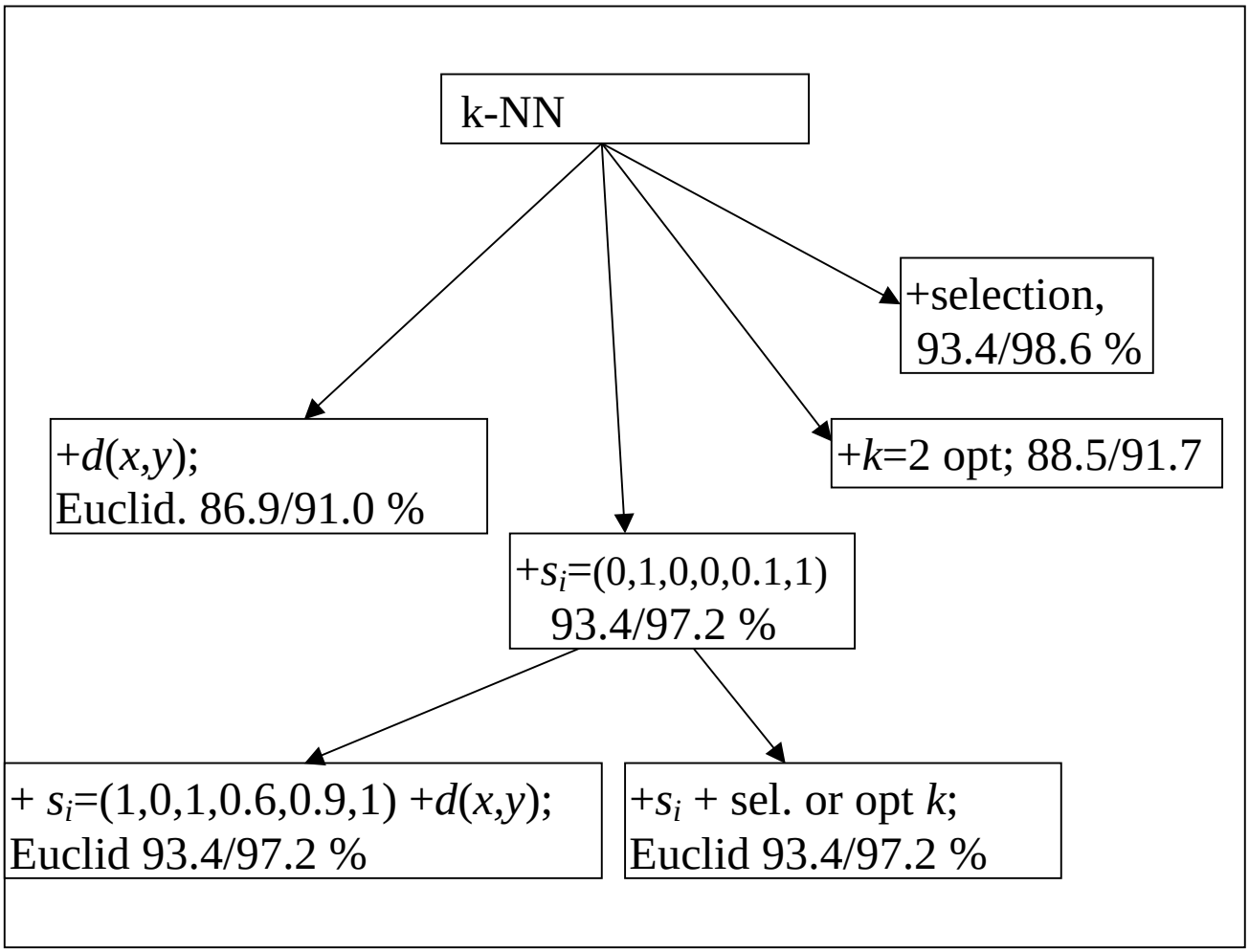
Best sequence: k-nn with Camberra distance function, training accuracy 89.9%, test accuracy 90.7%.

Monk-3 problem:

Rule: ¬ (body shape = octagon ∨ jacket color = blue) ∨
(holding = sward ∧ jacket color = green)

122 training cases, 432 test cases

6 mislabeled vectors in the training set (4.9%) to simulate the effects of noise in the data.



Although test results were better with feature selection only the training results may be used for evaluation!

Comparison of results for the Monk Problems

Inductive machine learning methods have some advantages for this type of problems, but SBL with meta-learning does fine.

	MONK-1	MONK-2	MONK-3 (%)
ML Methods			
ID3	**98.6**	**67.9**	**94.4**
ID5R	**79.7**	**61.8**	**95.2**
AQR	**95.9**	**79.7**	**87.0**
CN2	**100.0**	**69.0**	**89.1**
CLASSWEB 0.10	**71.8**	**64.8**	**80.8**
MLP+BP	**100**	**100**	**93.1**
SBL+meta	**100**	**90.7**	**97.2**

Hepatobiliary disorders.

Y. Hayashi, A. Imura, K. Yoshida, Fuzzy neural expert system and its pplication to medical diagnosis". 8th Int. Congress on Cybernetics and Systems, N.Y. City, pp. 54-61, 1990.
536 patients, Tokyo-based hospital

4 types of hepatobiliary disorders:

1. alcoholic liver damage (AL), 34.0%
2. primary hepatoma (PH), 23.9%
3. liver cirrhosis (LC), 22.3%
4. cholelithiasis (CH), 19.8%

Each record: 9 biochemical tests + sex.
163 cases used as the test data.

Strongly overlapping classes, difficult data.
100 fuzzy rules give about 75-76% accuracy.

k-nn, k=1, Euclidean distance, 72.7% L10 (77.9% test).

First level

Optimization of k, k=1, acc 72.7% (test 77.9%).
Optimization of d=Manhattan, 79.1% (test 77.9%).
Feature sel, removed 1 (Creatinine level), 74.3% (test 79.1%).
Feature weight, Euclidean, 78.0% (test 78.5%);
weights are [1.0, 1.0, 0.7, 1.0, 0.2, 0.3, 0.8, 0.8, 0.0].

New reference: Manhattan distance.

Second level:
k opt, k=1, no change
Feature sel, no change
Feature weighting, 80.1% (test 80.4\%).
(final weights are [1.0, 0.8, 1.0, 0.9, 0.4, 1.0, 1.0, 1.0, 1.0]

Method	Training set	Test set	Reference
IB2-IB4	81.2-85.5	44.6	WEKA, our calculation
Naive Bayes	--	46.6	WEKA, our
1R (rules)	58.4	50.3	WEKA, our
T2 (rules from decision tree)	67.5	53.3	WEKA, our
FOIL (inductive logic)	99	60.1	WEKA, our
FSM, 49 crisp logical rules	83.5	63.2	FSM, our
LDA (statistical)	68.4	65.0	our calculation
DLVQ (38 nodes)	100	66.0	our calculation
C4.5 decision rules	64.5	66.3	our calculation
Best fuzzy MLP model	75.5	66.3	Mitra et. al
MLP with RPROP		68.0	our calculation
Cascade Correlation		71.0	our calculation
Fuzzy neural network	100	75.5	Hayashi
C4.5 decision tree	94.4	75.5	our calculation
FSM, Gaussian functions	93	75.6	our calculation
FSM, 60 triangular functions	93	75.8	our calculation
IB1c (instance-based)	--	76.7	WEKA, our
kNN, k=1, Manhattan	79.1	77.9	our calculation, KG
K* method	--	78.5	WEKA, our
1-NN, 4 features removed, Manhattan	76.9	80.4	our calculation, KG

Ionosphere data

Data from UCI, 200 training vectors, 150 test.
34 continuous features, 2 classes.

Small, with many features, in the training set half from class1, in the test set only 18% from class 1.

k-nn, k=1, Euclidean distance, 86.0% (92.0% test).

First level
Optimization of k, k=1, acc 86.0% (test 92.0%).
Optimization of d=Manhattan, 87.5% (test 96.0%).
Feature sel, leaves 10 features, 92.5% (test 92.7%).
Feature weighting, Euclidean, 94.0% (test 87.3%);
  6 non-zero weights are left.

Second level, reference = feature weighting
Optimization of k, no change
Optimization of d=Manhattan, 95.0% (test 88.0%).
Feature sel, no change.

No correlation between results on the training and on the test set in this case!
Good results may be obtained by chance only.

StatLog preliminary results

23 algorithms & 22 datasets.
Goal: get best results using SBM for all datasets!

Data	k-NN rank	SBM rank	Best algorithm
Handwritten digits	1	1	k-NN
KL-digits	1	1	k-NN
SatImage	1	1	k-NN
Handwriting segmentation, Cut 50	1	1	k-NN
Image segmentation	16	1	Alloc80
Heart disease (+costs)	10	1	Bayes rule
German credit (+costs)	10	1	LDA
Australian credit	15	2	Cal5 (DT)
Cut 20	2	?	Bayes rule
Letters	2	?	Alloc80
Chromosome	5	?	QDA
Vehicle images	11	?	QDA

Conclusions

1. Meta-learning may automatically find the best combinations pf parameters and procedures escaping from the no-free-lunch curse.
2. Integration of many CI methods is possible in the SBM framework.
3. Many new CI methods result from the framework.
4. RBF, MLPs and other neural networks are also a special case of SBM.
5. Missing data, pattern completion, associative memory and other applications are possible.

Challenges:

- Programming all procedures/methods.
- Optimal search in the space of all models.
- SBL network realizations optimizing distance functions separately for each node.
- Approximation methods have not yet been included.