A few ML algorithms worth further development

Włodzisław Duch

Laboratorium Neurokognitywne,
Interdyscyplinarne Centrum Nowoczesnych Technologii UMK
Katedra Informatyki Stosowanej UMK

Google: Wlodzislaw Duch

Systemy inteligentne. teoria, praktyka, wyzwania, WMNI PW
21/11/2017

NeuroCog projects

Neurocognitive Informatics: understanding complex cognition => creating algorithms that work in similar way.

- Computational creativity, insight, intuition, imagery.
- Imagery agnosia, amusia, musical talent.
- Neurocognitive approach to language, word games.
- Brain stem models & consciousness in artificial systems.
- Medical information retrieval, analysis, visualization.
- Comprehensive theory of autism, ADHD, phenomics, education.
- Understanding neurodynamics, EEG signals, neurofeedback.
- Geometric theory of brain-mind processes.
- Infants: observation, perception/WM development.
- Neural determinism, free will & social consequences.

My group of neuro-cog-fanatics

In search of the sources of brain's cognitive activity

Project „Symfonia”, NCN, Kraków, 18 July 2016

FACULTY OF PHYSICS, ASTRONOMY AND INFORMATICS

CENTRE FOR MODERN INTERDISCIPLINARY TECHNOLOGIES

INSTITUTE OF PHYSIOLOGY AND PATHOLOGY OF HEARING

nencki institute of experimental biology

Computational QM

Lecture Notes in Chemistry

42

W. Duch

GRMS or Graphical Representation of Model Spaces

Springer-Verlag

Graphical representation of model spaces. Vol. I Basics.
Springer Verlag, Berlin Lecture Notes in Chemistry
Vol. 42 (1986); Vol. II has never been written ...

Idea:

Differential equations, such as Schrodinger equations in quantum mechanics are approximated by algebraic equations defined in tensor spaces with proper symmetry.

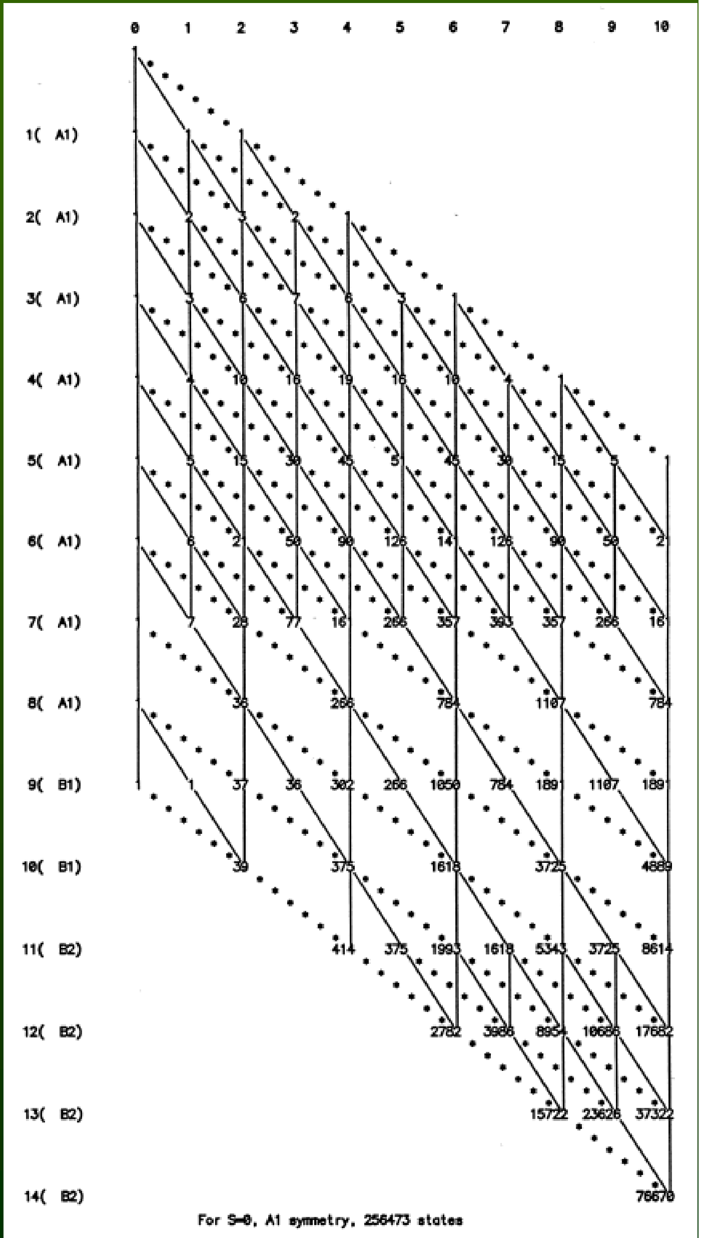
$$\hat{H}\Psi(x_1, x_2 \dots x_N) = E\Psi(x_1, x_2 \dots x_N) \rightarrow \mathbf{HC} = E\mathbf{C}$$

In finite dimensional N-particle Hilbert spaces eigenfunctions Ψ become linear combinations a large number of basis functions:

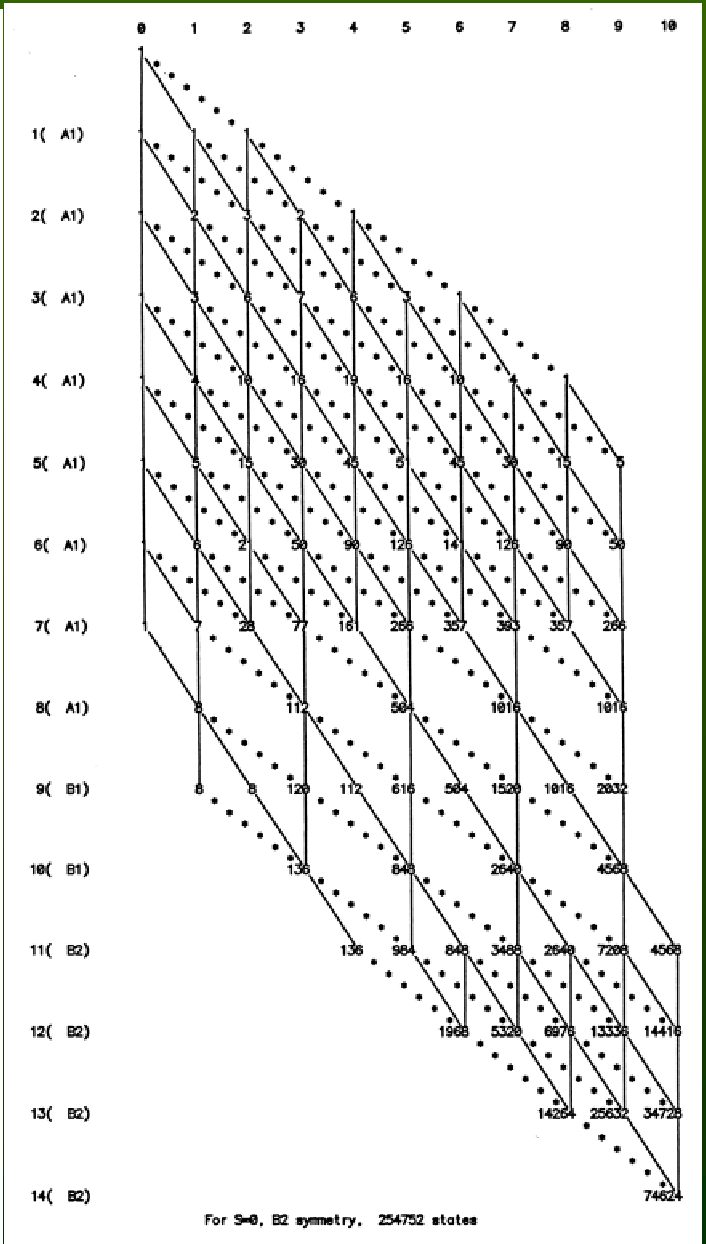
$$\Psi(x_1, x_2 \dots x_N) = \sum_I C_I \Phi_I (x_1, x_2 \dots x_N)$$

SGGA, Symmetric Group Graphical Approach; Lie algebra, Graphs.

Analyze graph to solve HC=EC



For S=0, A1 symmetry, 256473 states



For S=0, B2 symmetry, 254752 states

Discovering the wheel

1990-95 - before the Internet and repositories of papers …

RBF discovery:
Broomhead & Lowe (1988), Multivariable functional interpolation and adaptive networks. Complex Systems 2: 321–355.

Duch W (1994) Floating Gaussian Mapping: a new model of adaptive systems. Neural Network World 4:645-654

MDS: Torgerson (1958), Sammon 1969.

Duch W (1995) Quantitative measures for the self-organized topographical mapping. Open Systems and Information Dynamics 2:295-302 (a set of 3rd order equation instead of minimization).

Discovering the wheel - reverse

Although we have Internet some people have yet to rediscover my ideas …

- Duch W (1996) Computational physics of the mind. Computer Physics Communication 97: 136-15
- Perlovsky, L. I. (2016). Physics of the Mind. Frontiers in Systems Neuroscience, 10. fnsys.2016.00084

- Duch W, Diercksen GHF (1995) Feature Space Mapping as a universal adaptive system. Computer Physics Comm. 87: 341-371
- Perlovsky LI. Neural networks and intellect: Using model based concepts. New York: Oxford University Press; 2001.

Same with meta-learning, prototype-based learning, transfer functions, creativity and intuition, and a few other ideas.

- Cognitive informatics: HITs, DREAMs & Perfect Babies. A*STAR Cognitive Science Symposium, Singapore, September 26, 2005
- Duch W (2007), Intuition, Insight, Imagination and Creativity. IEEE Computational Intelligence Magazine 2(3), 40-52

Department of Informatics in 2008

CI Projects

Google W. Duch => List of projects, talks, papers

Computational intelligence (CI), main themes:

- Understanding of data: visualization, prototype-based rules.
- Foundations of computational intelligence: transformation based learning, k-separability, learning hard boole'an problems.
- Novel learning: projection pursuit networks, QPC (Quality of Projected Clusters), search-based neural training, transfer learning or learning from others (ULM), aRPM, SFM ...
- Similarity based framework for metalearning, heterogeneous systems, new transfer functions for neural networks.
- Feature selection, extraction, creation of enhanced spaces.
- General meta-learning, or learning how to learn, deep learning.

CI topics

1. Understanding NN functions
   - Visualization of network functions
   - Feature Space Mapping (FSM)
   - Prototype-rules (P-rules) and Fuzzy rules (F-rules)
2. NN learning algorithms
   - Support Vector Neural Training (SVNT)
   - Almost Random Projection Method (aRPM)
3. Foundations of CI
   - K-representability and complex logic
   - Quality of Projected Clusters (QPC)
   - Transformation-based learning, heterogeneous systems
   - Support Feature Machines (SFM) and Universal Learning Machines (ULM)
4. Meta-learning
   - Framework for Similarity-Based Learning (SBM)
   - Real Meta-Learning

Understanding NN functions

Visualization of NN mappings

Problem: NN are black boxes, dangerous to use. Can we trust NN solutions?

Partial solution: visualize error surfaces;
visualize activity of hidden/output neurons on known data+add some noise to investigate stability of NN mapping, convergence, compare different solutions, effects of regularization, type of functions/networks.

Visualization of multi-dimensional trajectories in weight space on error surfaces.

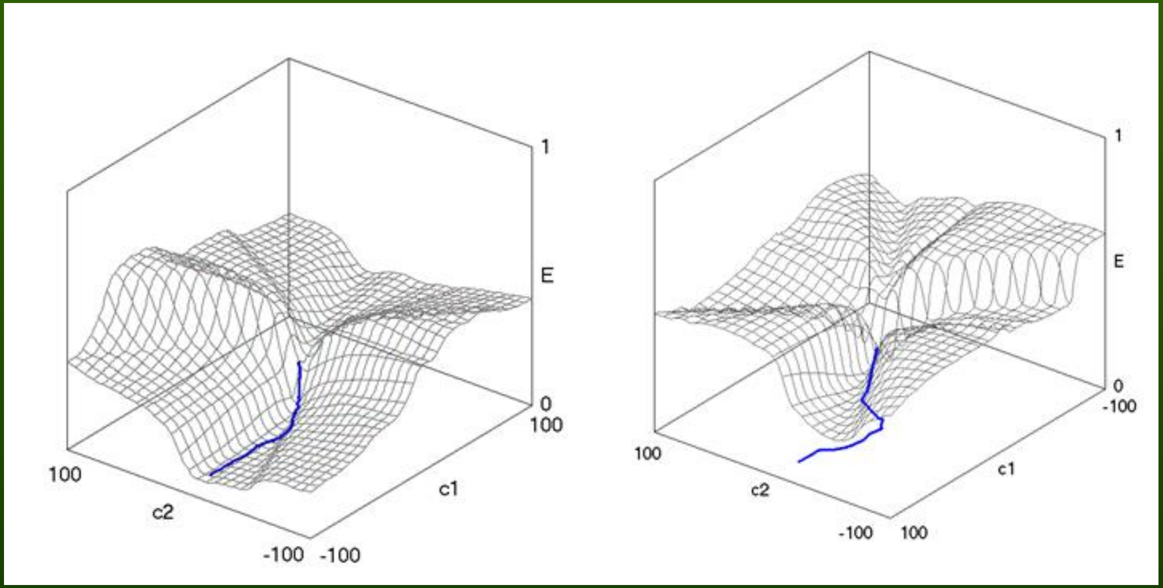
- Kordos, M., & Duch, W. (2005). A survey of factors influencing MLP error surface. Control and Cybernetics 33(4), 611–631.

Hidden secrets of neural networks. ICAISC Zakopane, Poland, June 2004

- Duch W, Visualization of hidden node activity in neural networks: I & II. Visualization methods. LN in AI 3070 (2004) 38-43; 44-49
- Duch W, Internal representations of multi-layered perceptrons. Issues in Intelligent Systems: Paradigms. 2005, pp. 49-62.

Trajectories of MLP weights

Take weights W_i from iterations i=1..K; do PCA on W_i covariance matrix. Captures 95-97% variance of the training data with just two vectors. Error functions in 2D PCA shows realistic learning trajectories!



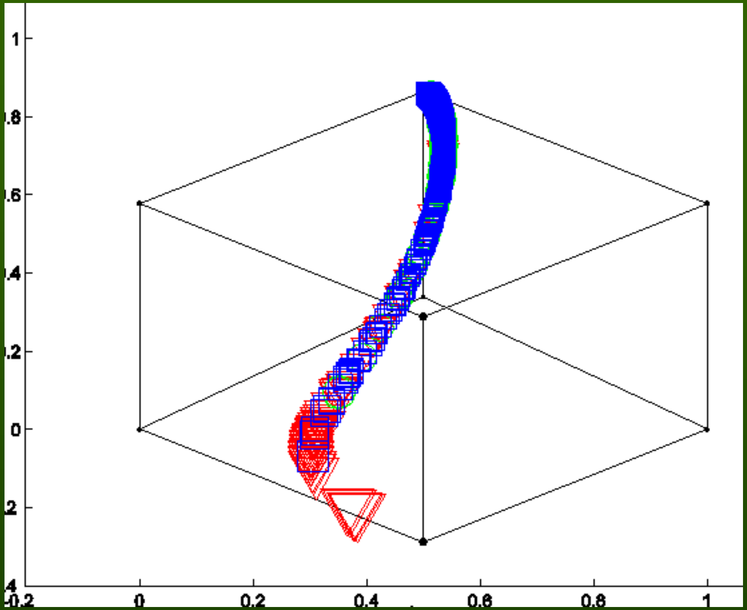
M. Kordos & W. Duch (2005)

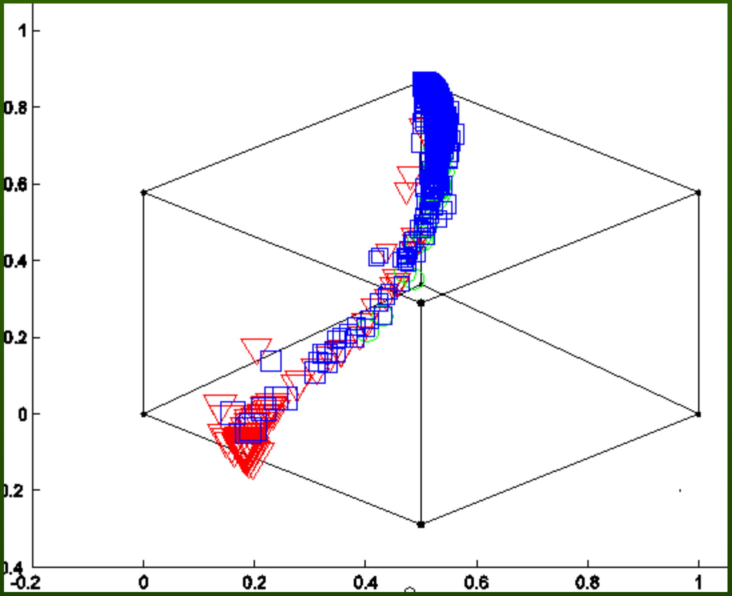
No local minima found. Many large flat plains and valleys.
Many initializations followed by short runs will find good solution.

Data far from decision borders has almost no influence, the main reduction of MSE is achieved by increasing ||W||, sharpening sigmoidal functions.

Dynamics of learning

Output images of the training data vectors mapped by NN:

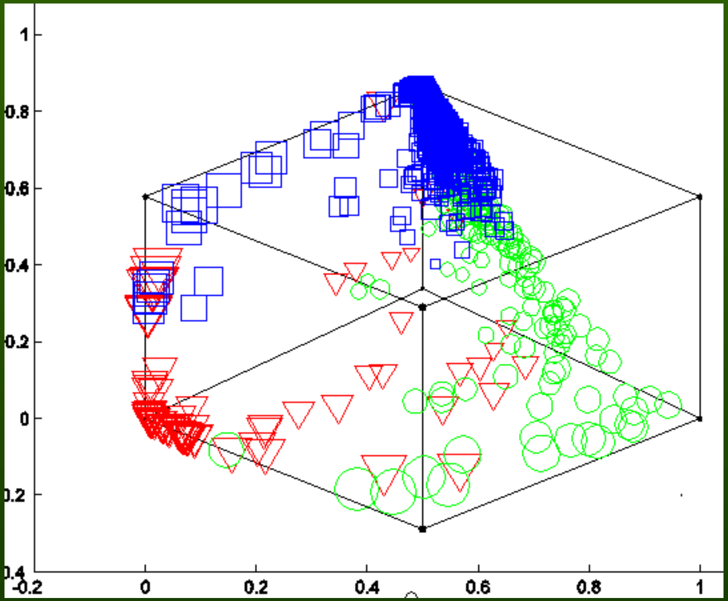


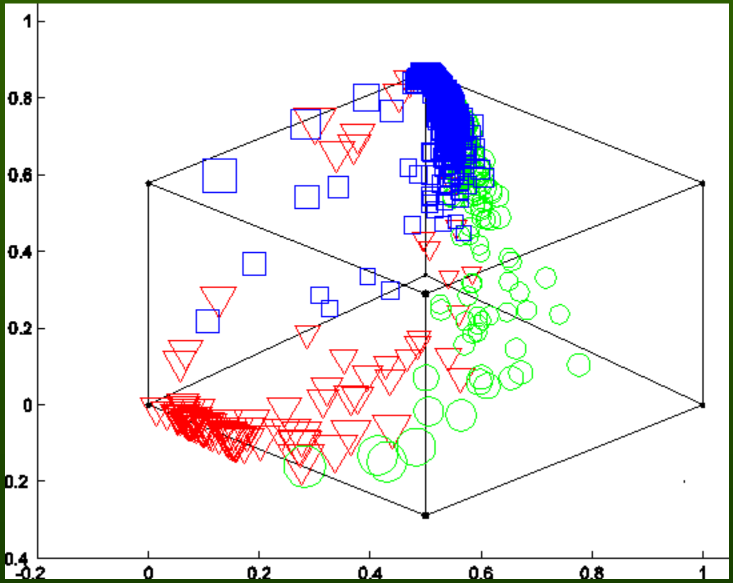


- Left: hypothyroid with 3 neurons the network will always under-fit the data, unable to separate small classes.
- Right: even if 6 hidden neurons are used problems with convergence may arise in some runs, but in other runs it may work.
- Stop further learning, start from another initialization.

Promising convergence

SCG training, network structure 21-8-3.
Output vectors displayed. Ideal results: (1,0,0), or (0,1,0) or (0,0,1).

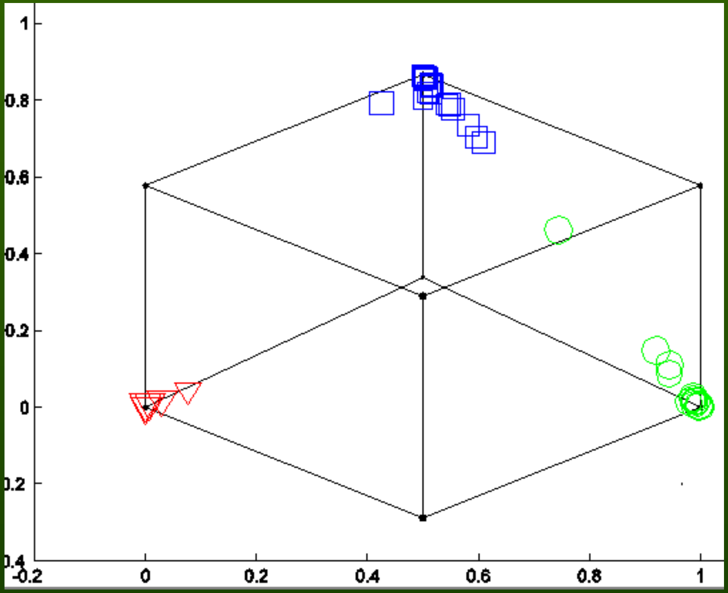


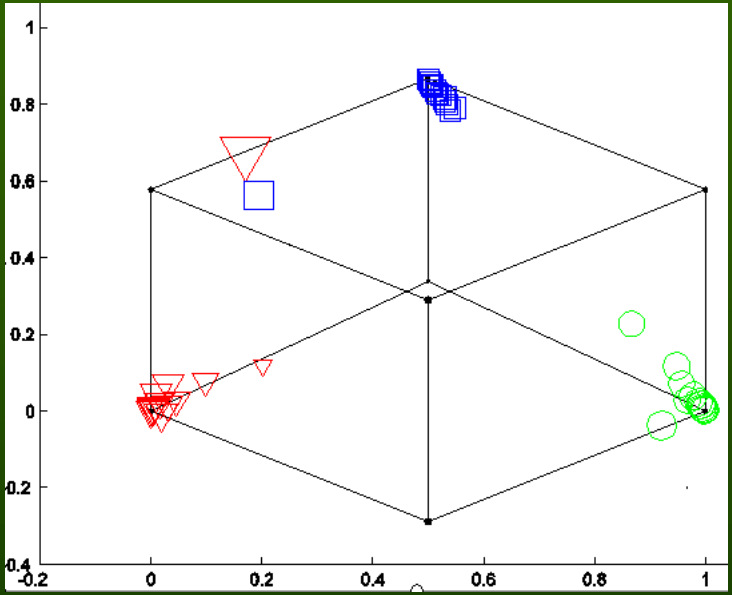


Two runs, intermediate solutions after 100 iterations,
left 164 right 197 errors.

Converged ...

Two very good converged solutions after 19 K and 22 K iterations, SCG training, 21-8-3, both cases 1 error, ~ same MSE.

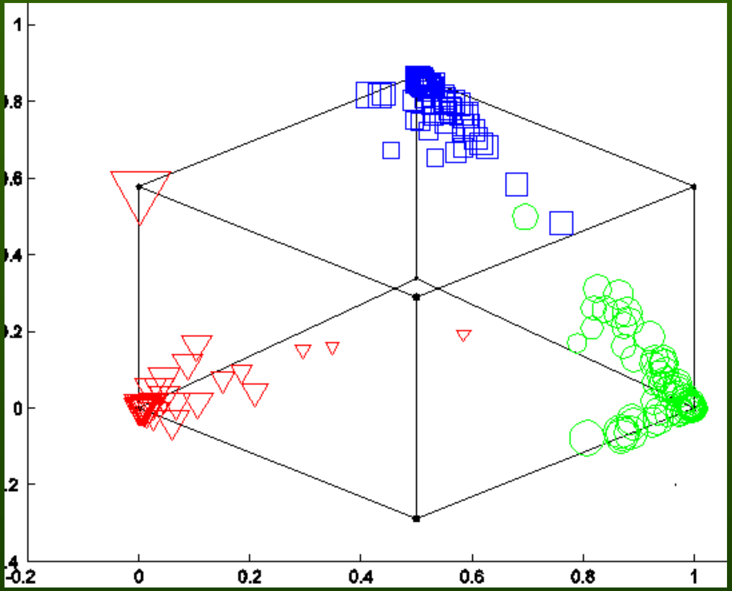


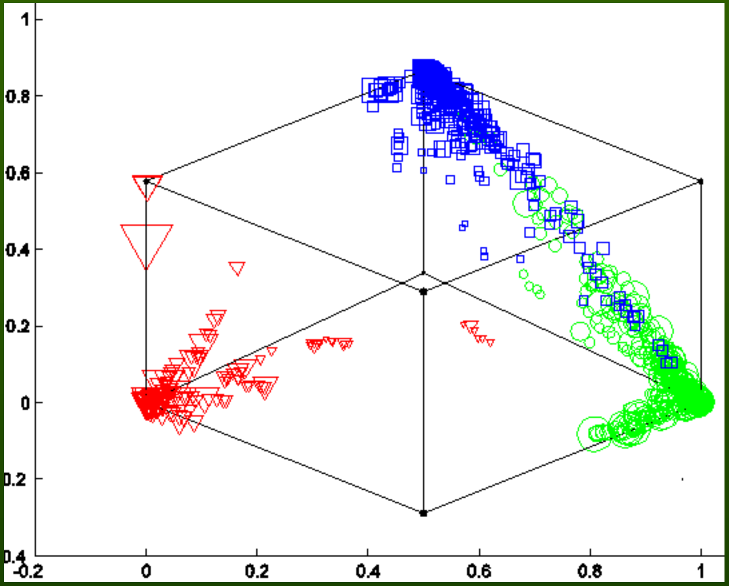


Left: green–blue potential mixing; right: blue–red classes mixing.
Networks are over-confident and not stable, weights are very large, sigmoids are step-like, decision borders are very sharp,
most training vectors are mapped to a single point, output is (1,0,0) etc.
New data near decision boundaries may fall on a wrong side.

Adding regularization

Regularization expands output clusters, but solution is more smooth and stable; below α=0.01 regularization is used.

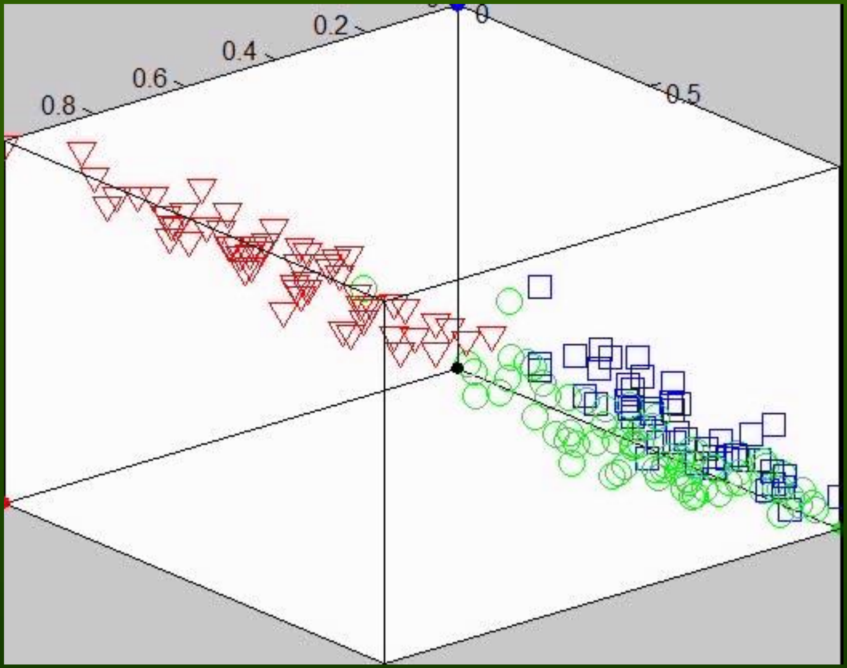


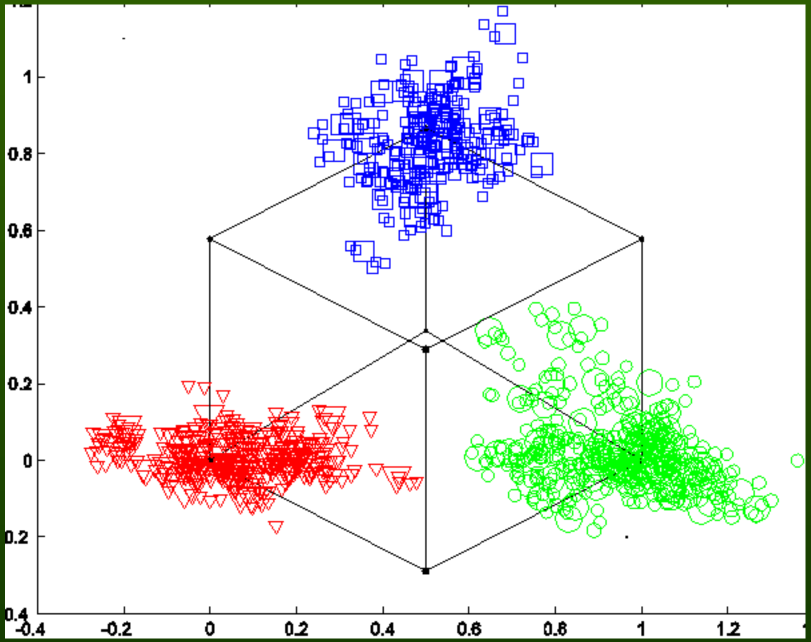


Left: solution with 12 hidden neurons, 2 errors;
Right: small variance (0.001) noise perturbation, 5 additional vectors per one training vector added => many errors appear.
Still regularization is too small.

MLP/RBF Wine solution

Wine data, 8-2-3 MLP and 8-6-3 RBF networks.
Perfect solutions may be dangerous! Add some noise to inputs to check for overfitting and see classification margins.





RBF always provides soft decision borders, not 0-1 outputs, but still separable clusters, more stable when small perturbations are added.

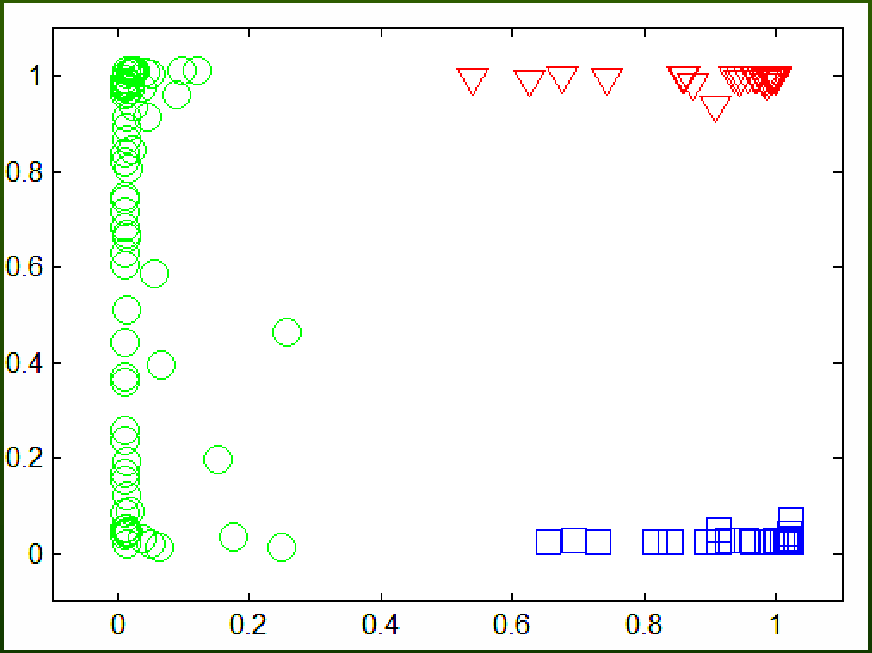
What is inside?

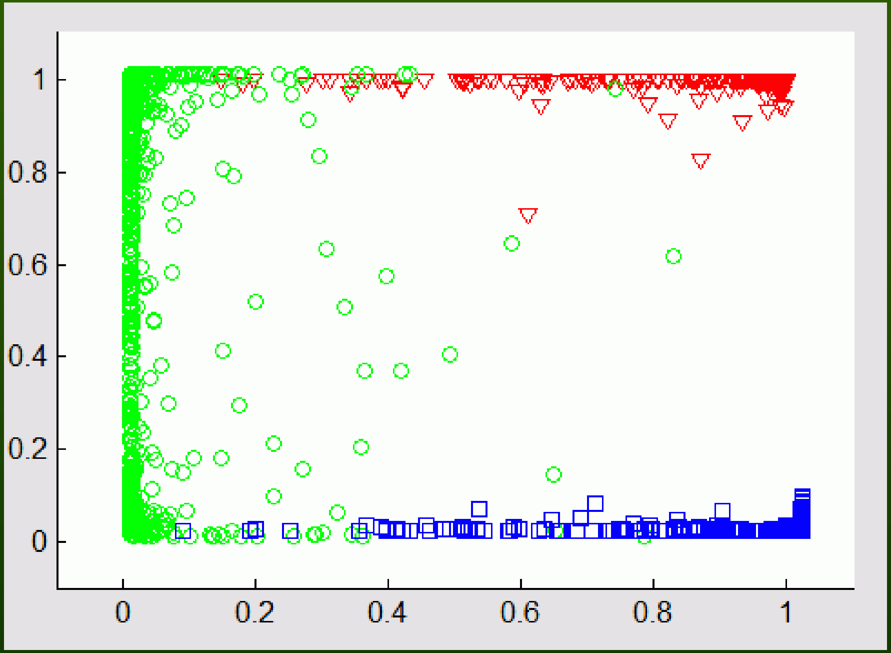
Many types of internal representations may look identical from outside, but generalization may depend on them.

- Classify different types of internal representations.
- Take permutational invariance into account: equivalent internal representations may be obtained by re-numbering hidden nodes.
- Good internal representations should form compact clusters in the internal space.
- Check if the representations form separable clusters.
- Discover poor representations and stop training.
- Analyze adaptive capacity of networks.
- .....

Wine: hidden 2D

Many solutions in 2D hidden space: which is the best?
Add noise to input data to see how stable the solution is.
Here each X => 10 points X+Gaussian noise with 0.05 variance.

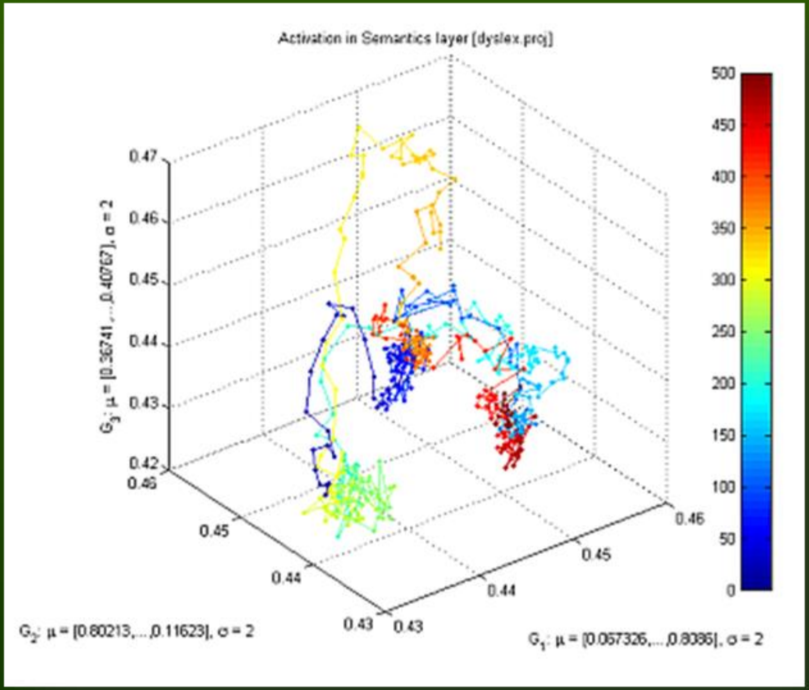


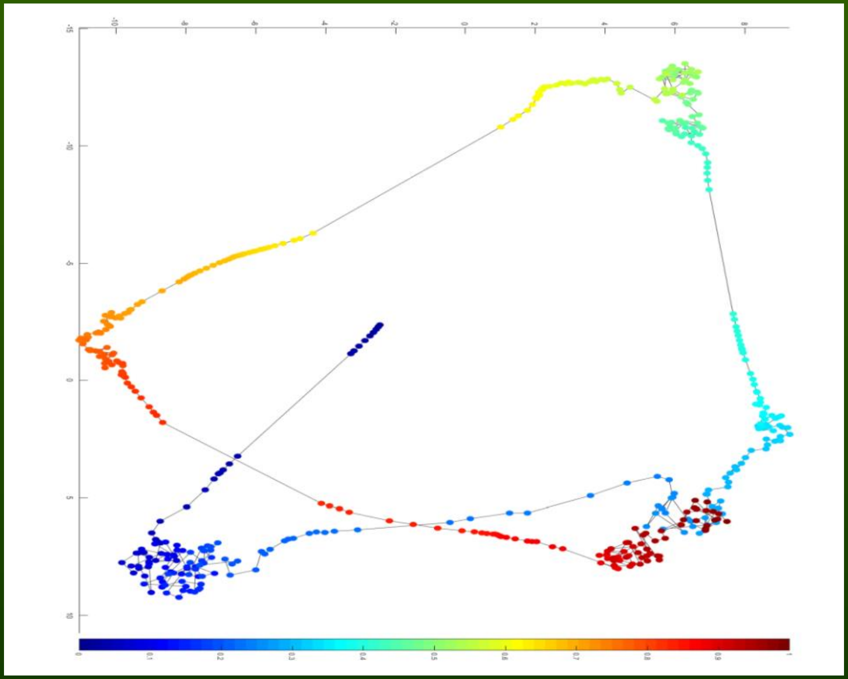


Analysis n higher dimensions done using parallel coordinates, projections to polyhedra, FSD, SNE and other techniques.

Hi-D trajectories

140 active units (dyslexia model in Emergent neural simulator); noisy attractor dynamics, FSD and SNE trajectories with 500 points.





Thanks to K. Dobosz Viser toolbox, and M. Orliński SNE implementation.

What feedforward NN really do?

Vector mappings from the input space to hidden space(s), and finally to the output space where data should be separable.

Hidden-Output mapping done usually by perceptrons.

A single hidden layer case is analyzed below.

$T = \{X^i\}$ training data, N-dimensional.

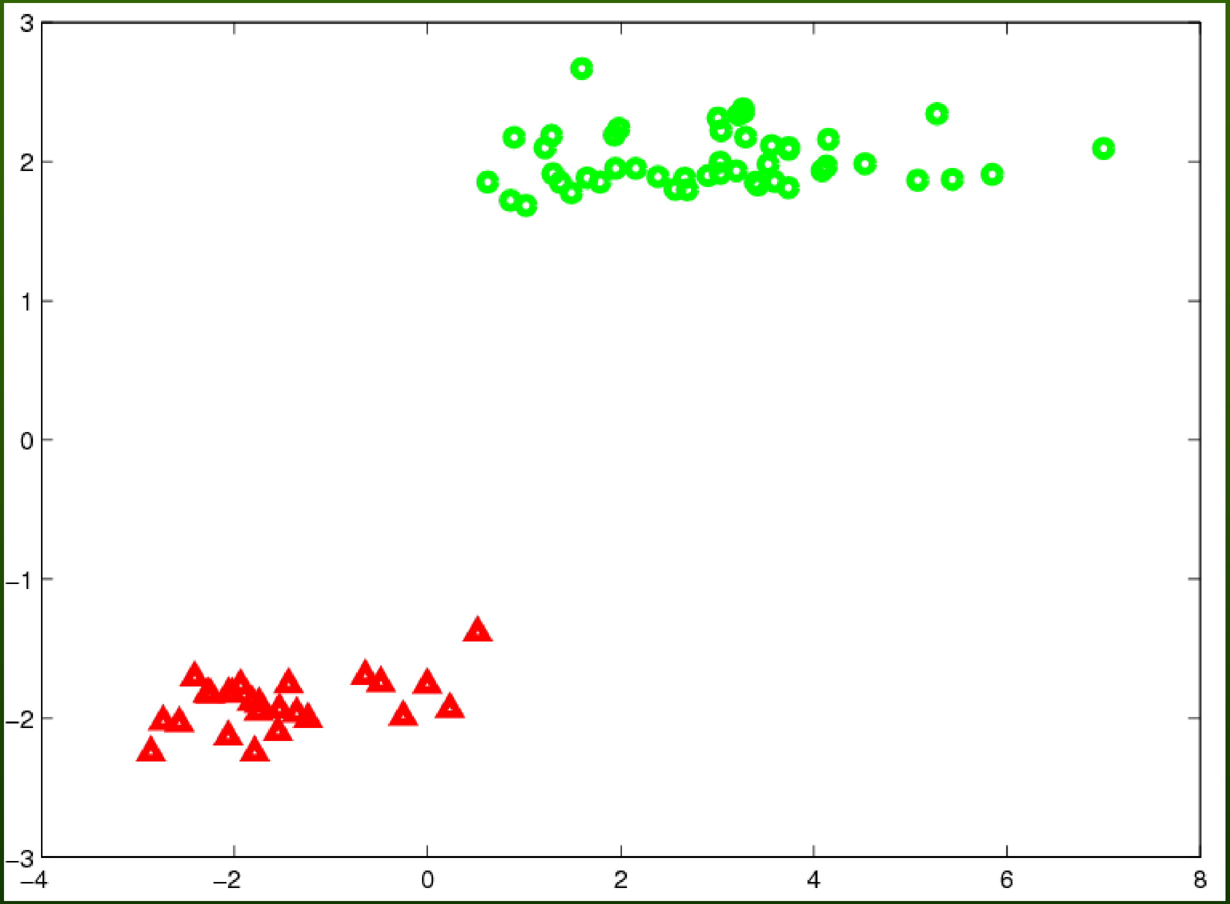
$H = \{h_j(X^i)\}$ T image in the hidden space, $j = 1$ .. N_H-dim.

$Y = \{y_k\{h(X^i)\}$ T image in the output space, $k = 1$ .. N_C-dim.

NN goal: scatterograms of H, the image of T in the hidden space should be linearly separable; internal representations will determine network generalization capabilities and other properties.

Is this a good goal? Can it be easily achieved?
"Universal approximator" theorem is not helpful.

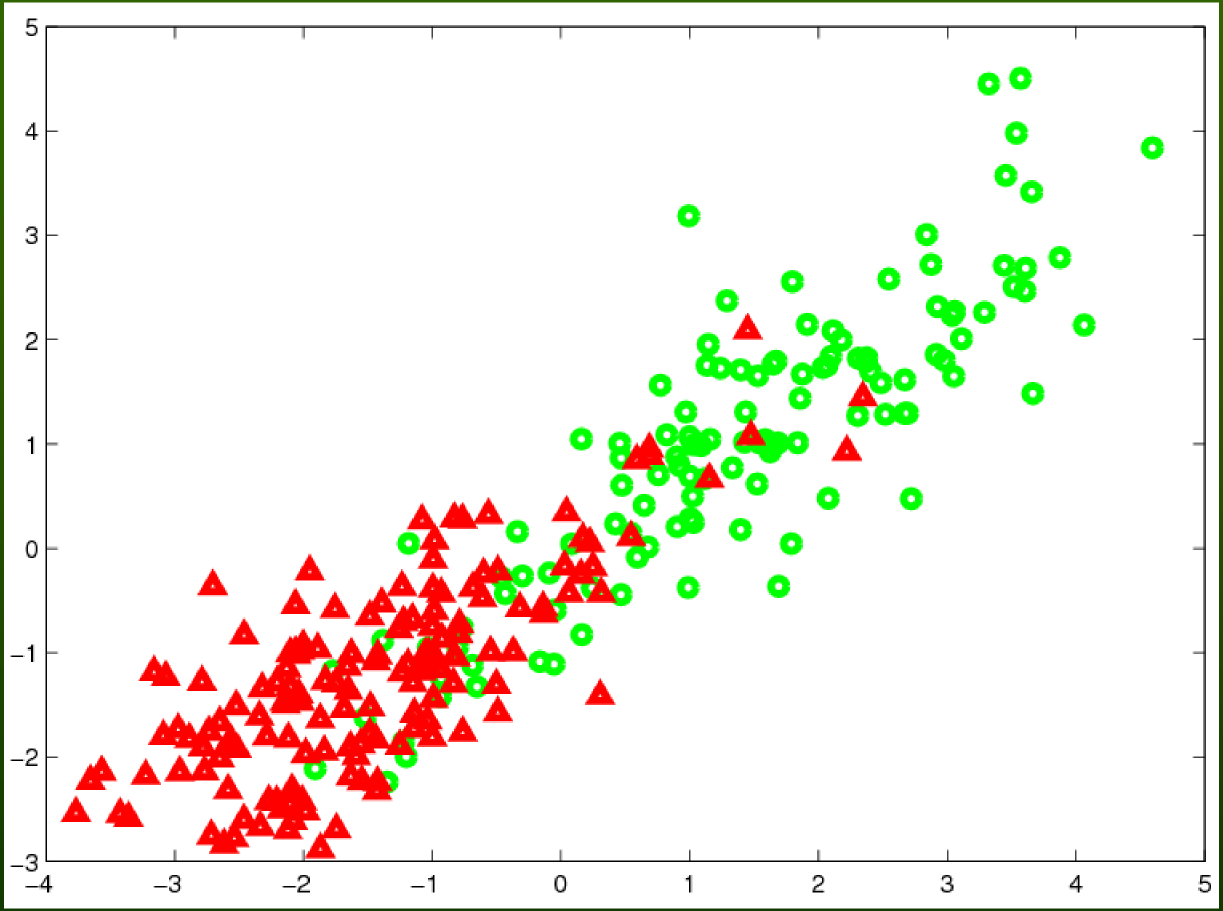
Linear separability



SVM visualization of Leukemia microarray data,

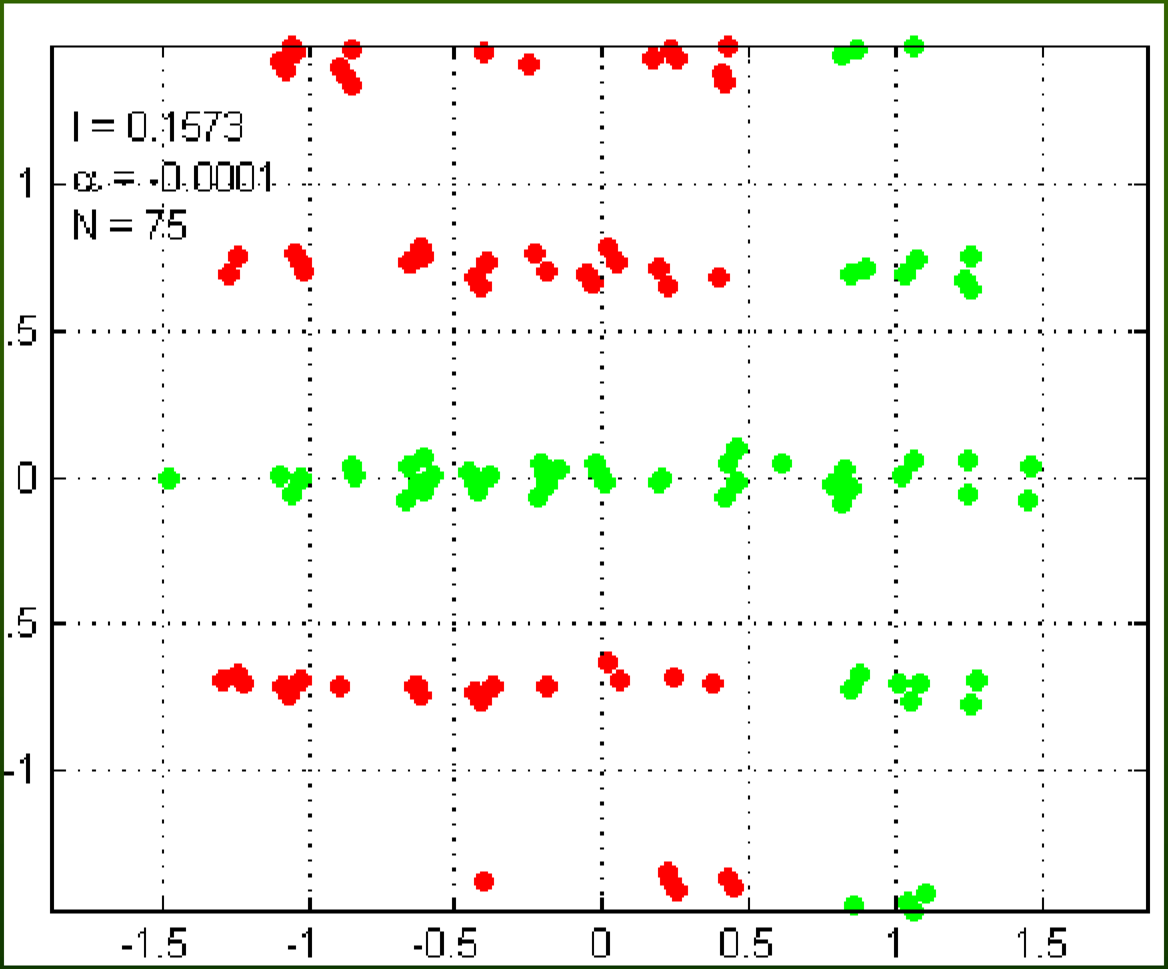
Horizontal axis x=WX, vertical - orthogonal projection.

Approximate separability



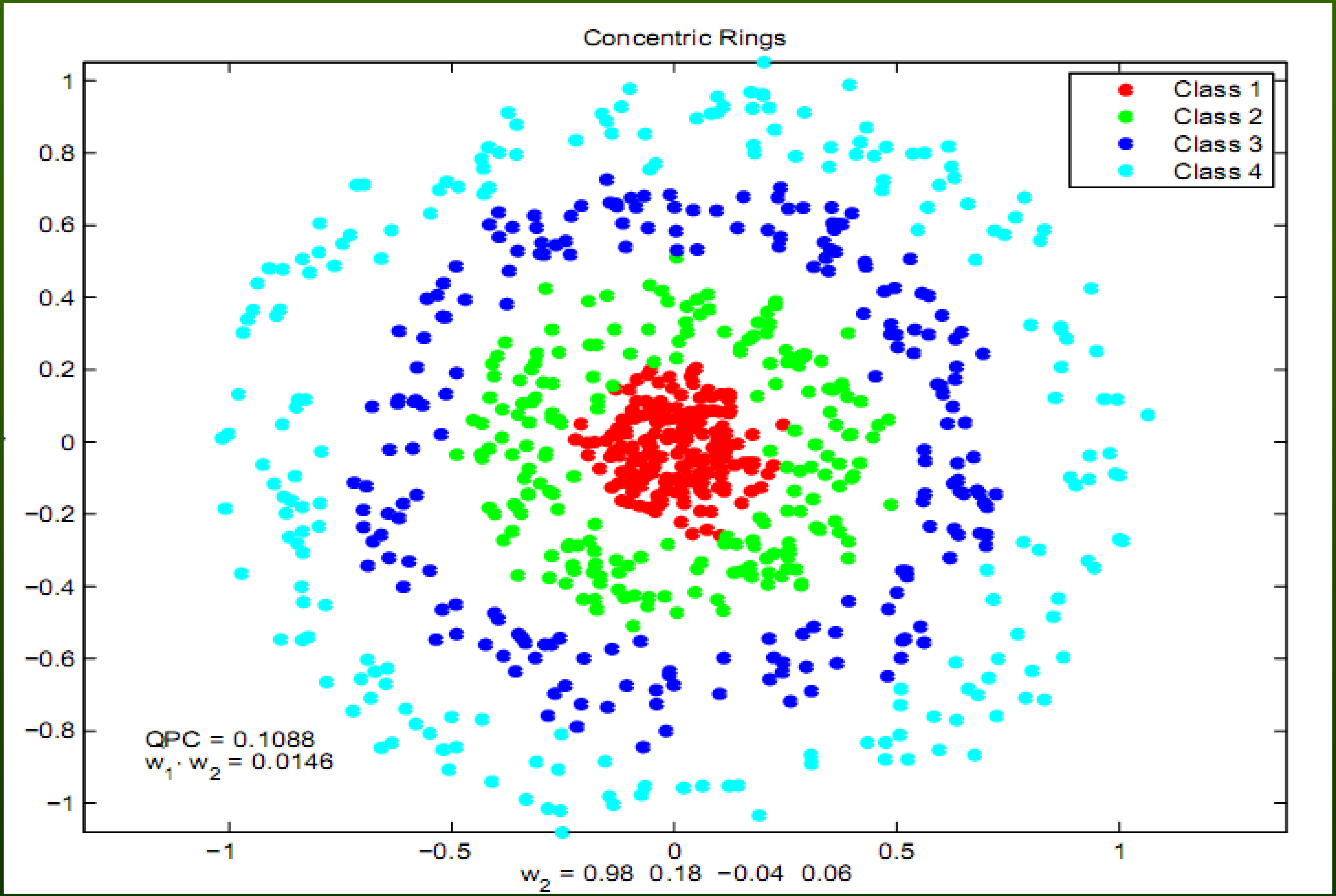
SVM visualization of Heart dataset, overlapping clusters, information in the data is insufficient for perfect classification.

Rules



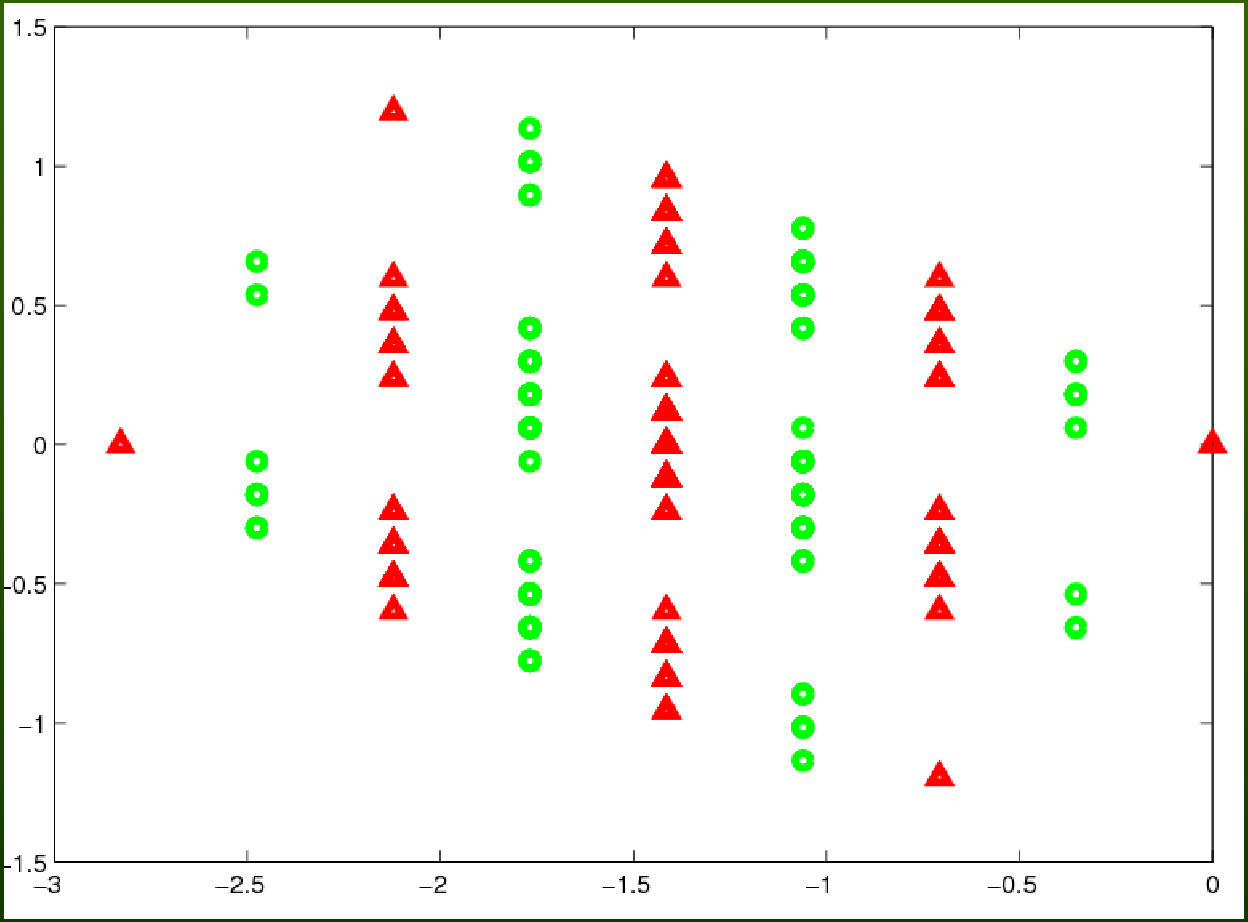
QPC visualization of Monks dataset with simple logical structure, two logical rules are needed, or combination of two projections.

Complex distribution



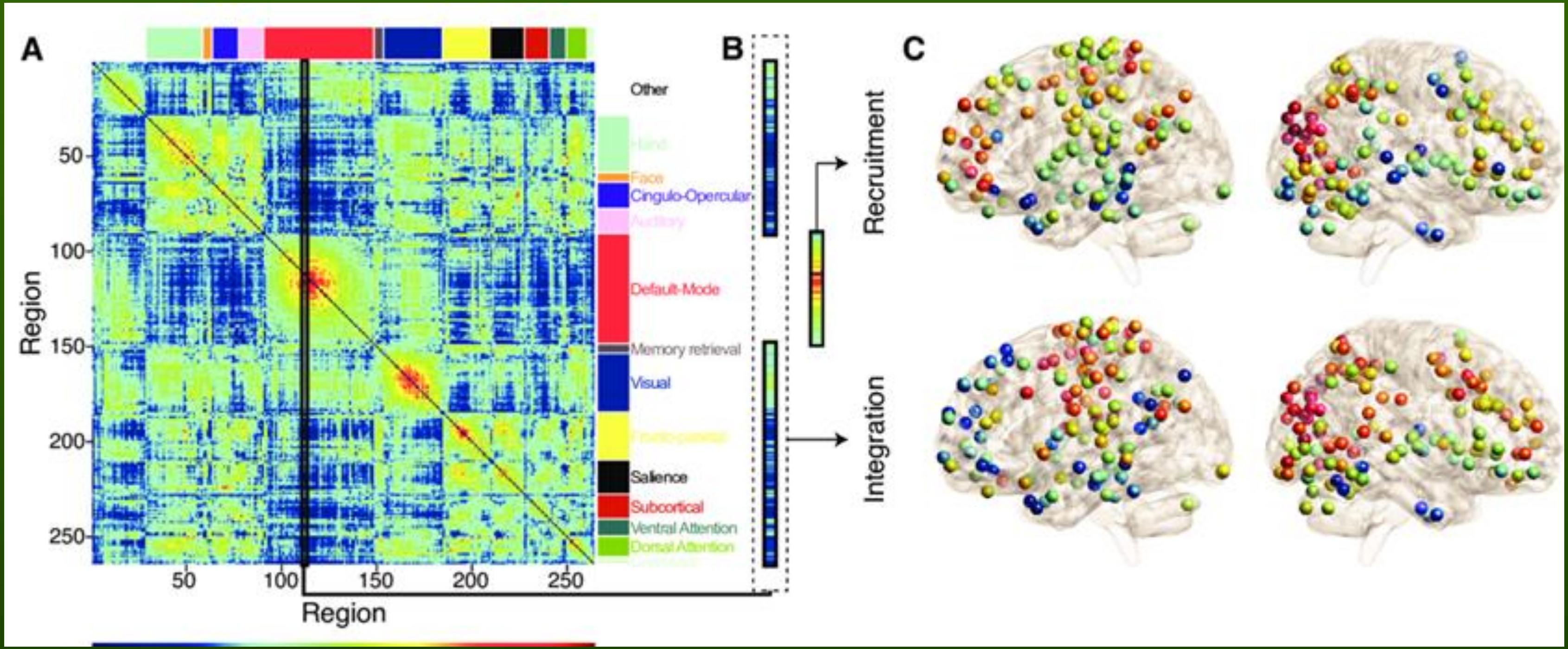
QPC visualization of concentric rings in 2D with strong noise in remaining 2D; transform: nearest neighbor solutions, combinations of ellipsoidal densities.

Interval transformation



8-bit parity data: 9-separability is much easier to achieve than full linear separability; almost impossible to train MLP on such data.

Much more complex logic …



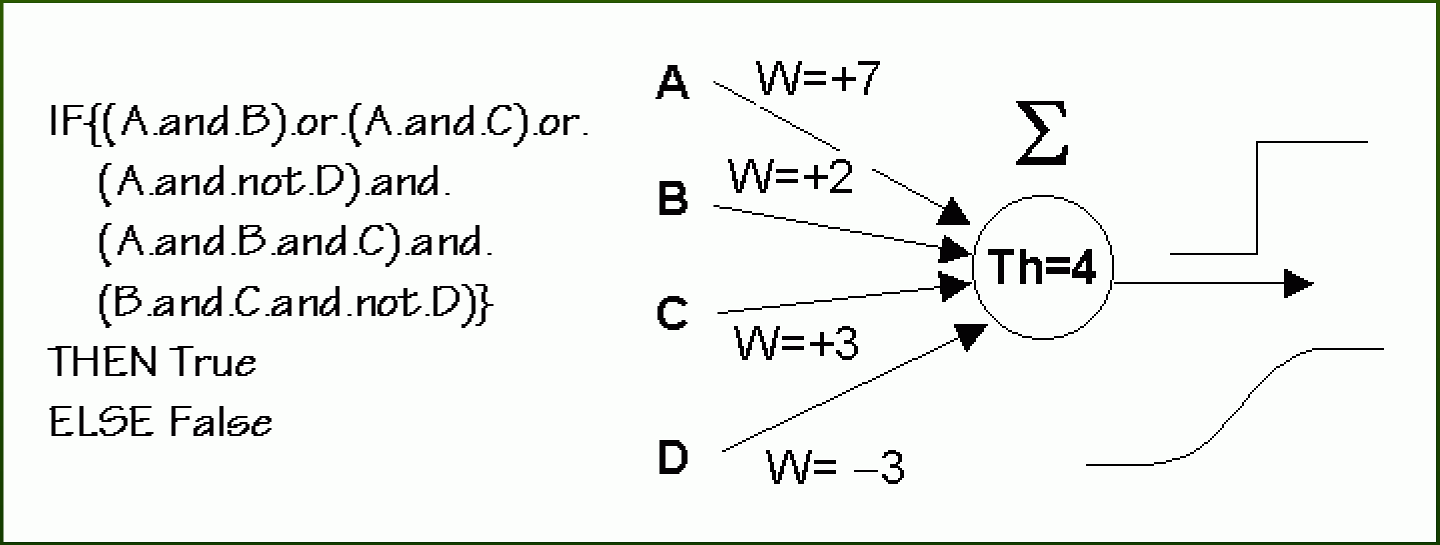
Different activations => same cognitive functions in different context? Complex logic may be needed to learn from this type of data.

Mattar, M. G., Cole, M. W., Thompson-Schill, S. L., & Bassett, D. S. (2015). A **Functional Cartography of Cognitive Systems**. *PLOS Computational Biology*, *11*(12), e1004533.

Rules from MLPs

Why is it difficult?

Multi-layer perceptron (MLP) networks: stack many perceptron units, performing threshold logic:
M-of-N rule: IF (M conditions of N are true) THEN ...

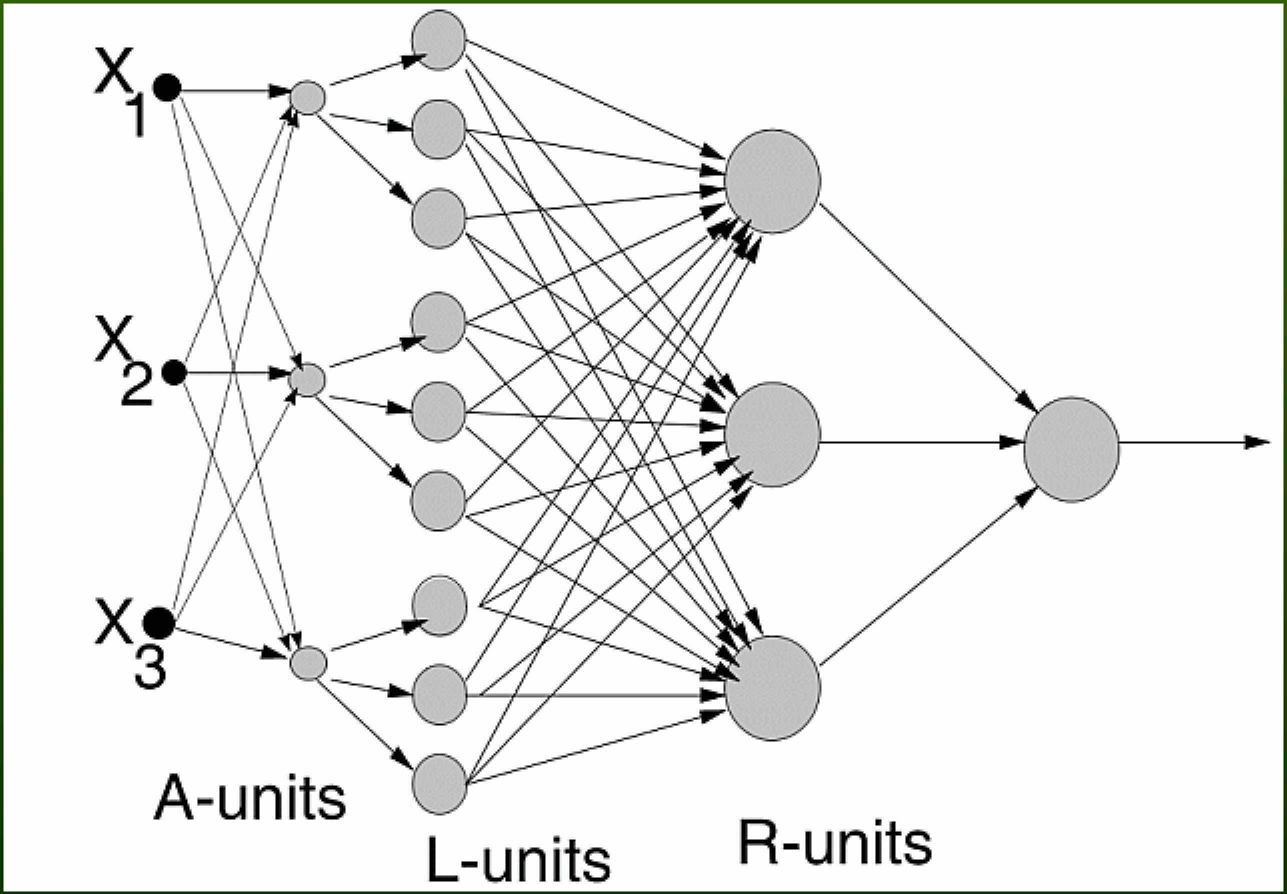


Problem: for N inputs number of subsets is 2^N.
Exponentially growing number of possible conjunctive rules.

MLP2LN

Converts MLP neural networks into a network performing logical operations (LN).

Input layer



Output: one node per class.

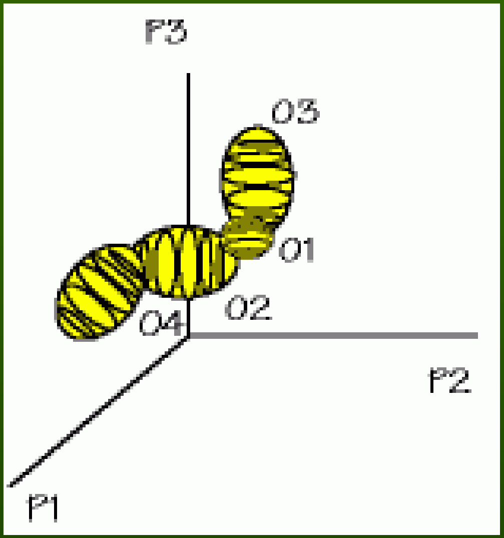
Aggregation: better features

Linguistic units: windows, filters

Rule units: threshold logic

FSM - neurofuzzy systems

Feature Space Mapping (FSM) constructive neurofuzzy system. Neural adaptation, estimation of probability density distribution (PDF) using single hidden layer network (RBF-like), with nodes realizing separable basis functions (SBF networks):



$$RBF\left(X;P\right)=\sum_i W_i \left\| X_i - P_i \right\|$$

$$FSM\left(X;P\right)=\sum_i W_i \prod_{j=1} G_{ij}\left(X_{ij} - P_{ij}\right)$$

Model of mental processes–FSM nodes representing attractors, mental events.

Duch W, Diercksen GHF (1995) Feature Space Mapping as a universal adaptive system. Computer Physics Communications 87: 341-371

Duch W (1997) Platonic model of mind as an approximation to neurodynamics. In: Brain-like computing and intelligent information systems, ed. S-i. Amari, N. Kasabov (Springer, Singapore 1997), chap. 20

Prototype-based rules

C-rules (Crisp), are a special case of F-rules (fuzzy rules).
F-rules (fuzzy rules) are a special case of P-rules (Prototype)!
P-rules have the form:

IF P = arg $\min_R$ D(X,R) THAN Class(X)=Class(P)

D(X,R) = dissimilarity (distance) function, determining local decision borders.

P-rules are easy to interpret and offer better description than F-rules!

IF X=You are most similar to the P=Superman
THAN You are in the Super-league.

“Similar” may involve different features or D(X,P). What is similar for the brain?
Kernel features in SVM are particular example of similarity functions.

- Duch W, Setiono R, Zurada J.M, Computational intelligence methods for understanding of data. Proc. of the IEEE 92(5) (2004) 771- 805
- Duch W, Adamczak R, Grąbczewski K, A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. IEEE Transactions on Neural Networks 12 (2001) 277-306

P-rules

Euclidean distance leads to a Gaussian fuzzy membership functions + product as T-norm. In this case FSM = RBF.

$$D(\mathbf{X},\mathbf{P}) = \sum_i d(X_i, P_i) = \sum_i W_i (X_i - P_i)^2$$

$$\mu_P(\mathbf{X}) = e^{-D(\mathbf{X},\mathbf{P})} = e^{-\sum_i d(X_i,P_i)} = \prod_i e^{-W_i(X_i-P_i)^2} = \prod_i \mu_i(X_i, P_i)$$

Manhattan function => $\mu(X;P)=\exp\{-|X-P|\}$

Various distance functions lead to different MF.
Ex. data-dependent probabilistic distance functions for symbolic data:

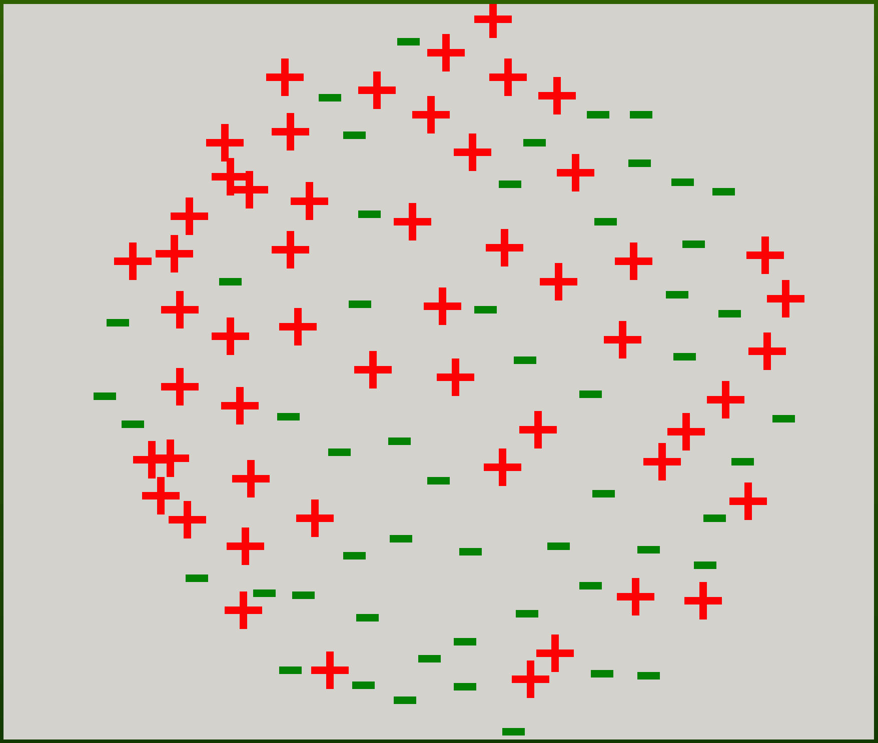
$$D_{VDM}(\mathbf{X},\mathbf{Y}) = \sum_i \left[\sum_j \left| p(C_j \mid X_i) - p(C_j \mid Y_i) \right| \right]$$

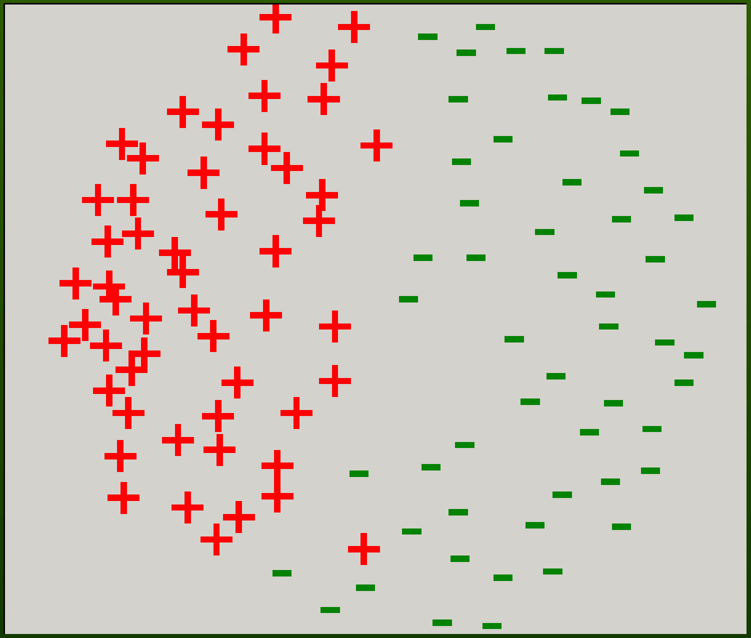
$$D_{PDF}(\mathbf{X},\mathbf{Y}) = \sum_i \left[\sum_j \left| p(X_i \mid C_j) - p(C_j \mid Y_i) \right| \right]$$

Promoters with p-distances

DNA strings, 57 aminoacids, 53 + and 53 - samples

tactagcaatacgcttgcgttcggtggttaagtatgtataatgcgcgggcttgtcgt





Euclidean distance, symbolic
s =a, c, t, g replaced by x=1, 2, 3, 4

PDF distance, symbolic
s=a, c, t, g replaced by p(s|+)

P-rules and C-rules

New distance functions from info theory => interesting MF.
MF => new distance function, with local similarity D(X,R) for each cluster.

Crisp logic rules: use Chebyshev distance (L_∞ norm):

$$D_{Ch}(\mathbf{X},\mathbf{P}) = \|\mathbf{X}-\mathbf{P}\|_\infty = \max_i W_i\,|X_i-P_i|$$

$D_{Ch}(\mathbf{X},\mathbf{P}) = \text{const}$ => rectangular contours.

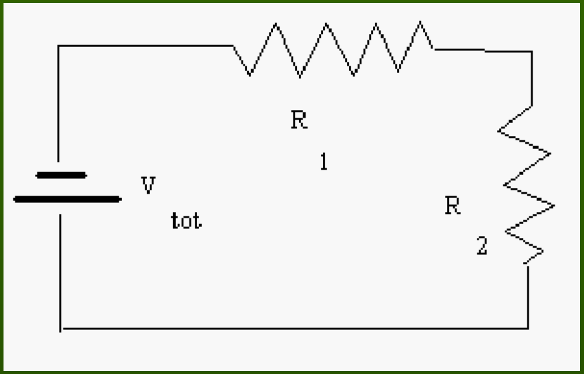
Chebyshev distance with thresholds θ_P

IF $D_{Ch}(\mathbf{X},\mathbf{P}) \le \theta_P$ THEN $C(\mathbf{X})=C(\mathbf{P})$

is equivalent to a conjunctive crisp rule

IF $X_1 \in [P_1-\theta_P/W_1, P_1+\theta_P/W_1] \wedge \ldots X_N \in [P_N-\theta_P/W_N, P_N+\theta_P/W_N]$
THEN $C(\mathbf{X})=C(\mathbf{P})$

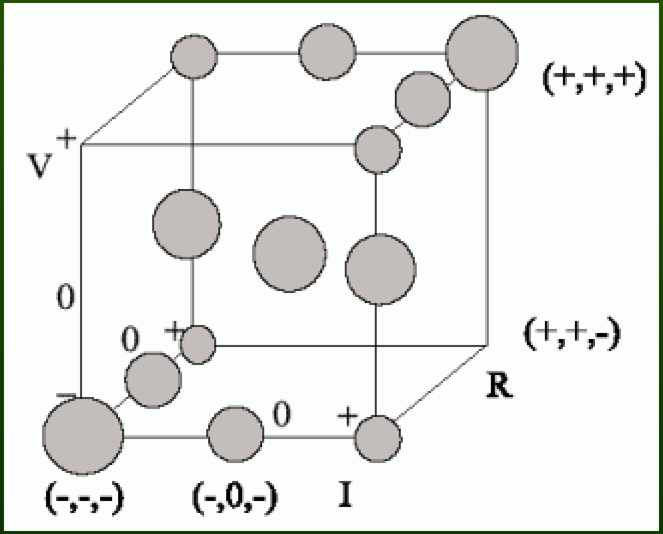
P-rules and intuitive thinking



Question in qualitative physics (PDP book): if R_2 increases, R_1 and V_t are constant, what will happen with current and V_1, V_2 ?

Learning from partial observations:

Ohm's law $V=I\times R$; Kirhoff's $V=V_1+V_2$.



Geometric representation of qualitative facts:

+ increasing, 0 constant, - decreasing.

True (I_-,V_-,R_0), (I_+,V_+,R_0), false (I_+,V_-,R_0).

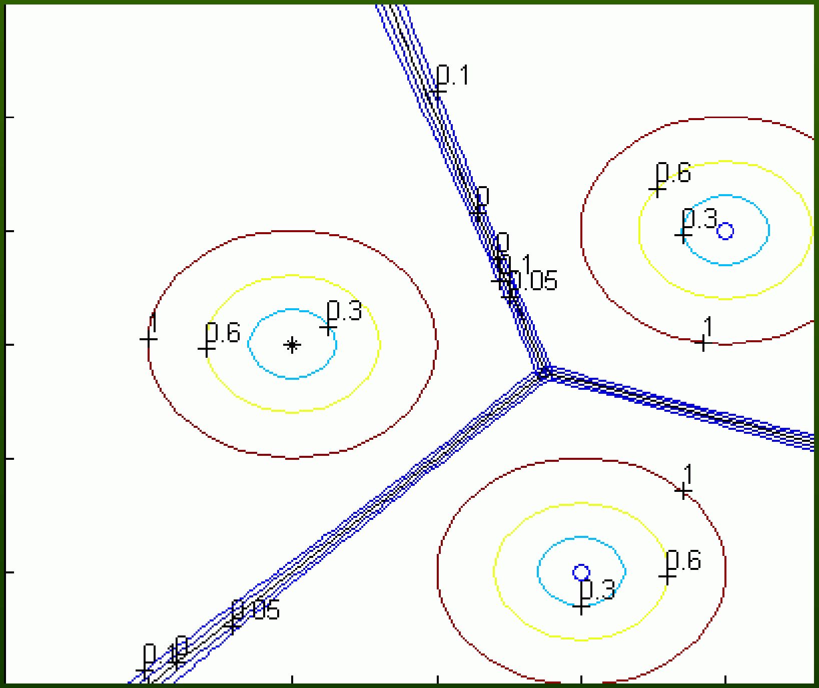
5 laws: 3 Ohm's   2 Kirhoff's laws.

All laws A=B+C, A=B×C , $A^{-1}=B^{-1}+C^{-1}$, have identical geometric interpretation!

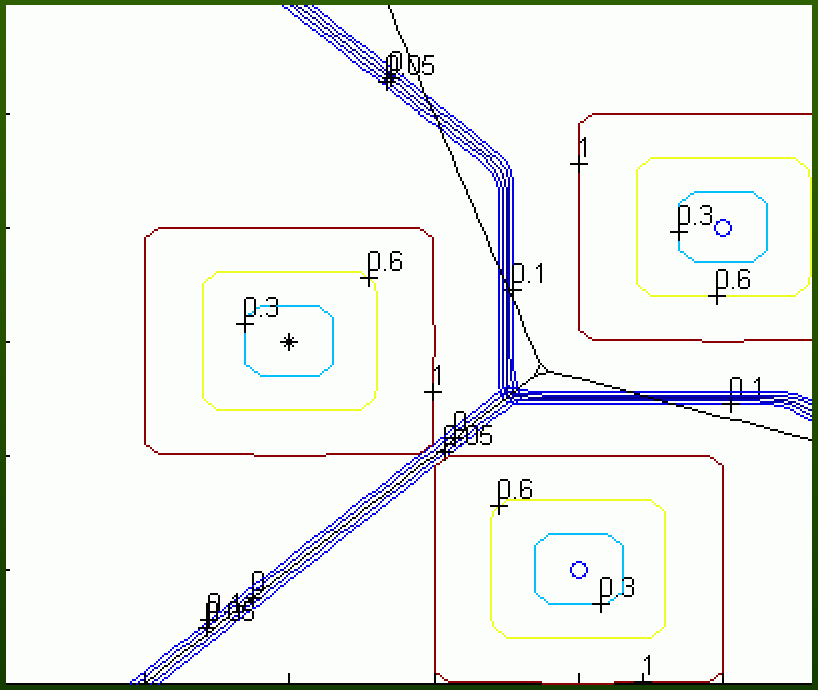
13 true, 14 false facts; simple P-space, but complex neurodynamics.

Decision borders

$D(P,X)=$const and decision borders $D(P,X)=D(Q,X)$.



Euclidean distance from 3 prototypes, one per class.

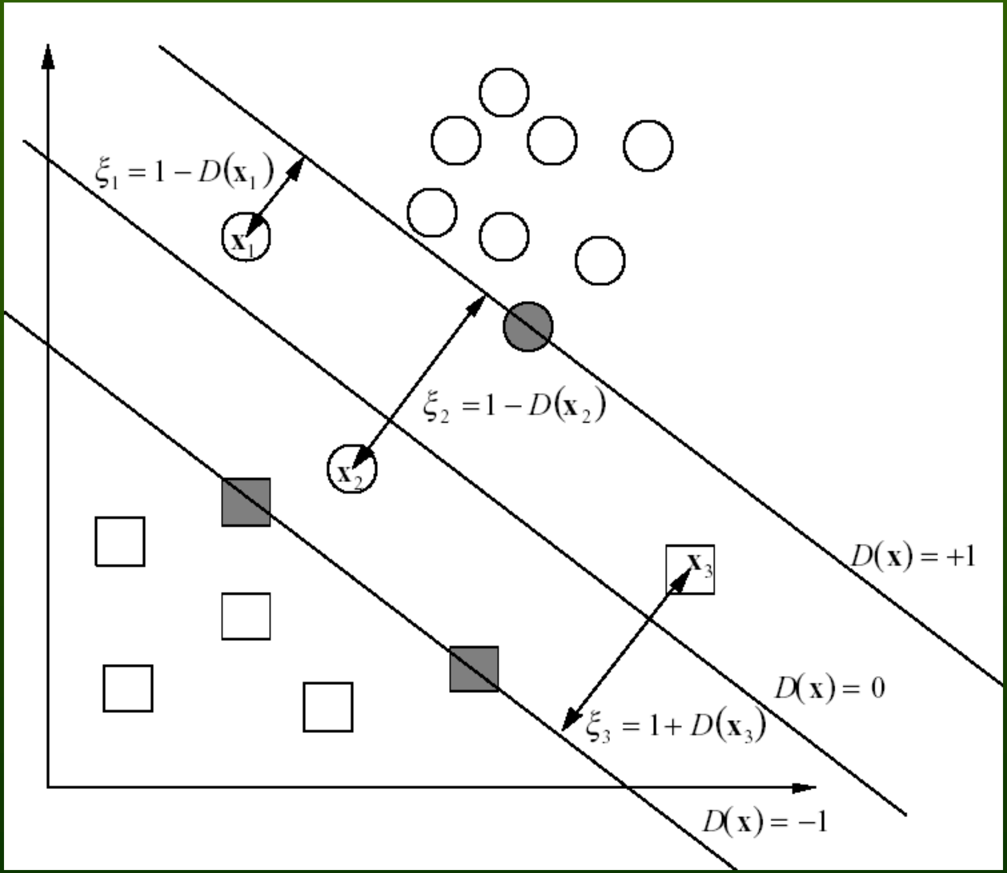


Minkovski $\alpha=20$ distance from 3 prototypes.

NN learning algorithms

Support Vectors

SVM gradually focuses on the training vectors near the decision hyperplane – can we do the same with MLP?



Selecting Support Vectors

Active learning: if contribution to the parameter change is negligible remove the vector from training set.

$$\Delta W_{ij} = -\eta \frac{\partial E(\mathbf{W})}{\partial W_{ij}} = -\eta \sum_{k=1}^{K} \left(Y_k - M_k(\mathbf{X};\mathbf{W})\right)^2 \frac{\partial M_k(\mathbf{X};\mathbf{W})}{\partial W_{ij}}$$

If the difference $\varepsilon_{\mathbf{W}}(\mathbf{X}) = \eta \sum_{k=1}^{K} \left| Y_k - M_k(\mathbf{X};\mathbf{W}) \right| \le \varepsilon_{\min}$

is sufficiently small the pattern X will have negligible influence on the training process and may be removed from the training.

Conclusion: select vectors with $\varepsilon_{W}(X)>\varepsilon_{min}$, for training.

2 problems: possible oscillations, and strong influence of outliers.

Solution: adjust ε_{min} dynamically to avoid oscillations;
remove also vectors with $\varepsilon_{W}(X) > 1-\varepsilon_{min} = \varepsilon_{max}$

SVNT algorithm (2005)

Initialize the network parameters W,

set $\Delta\varepsilon$=0.01, ε_{min}=0, set SV=T.

Until no improvement is found in the last N_{last} iterations do

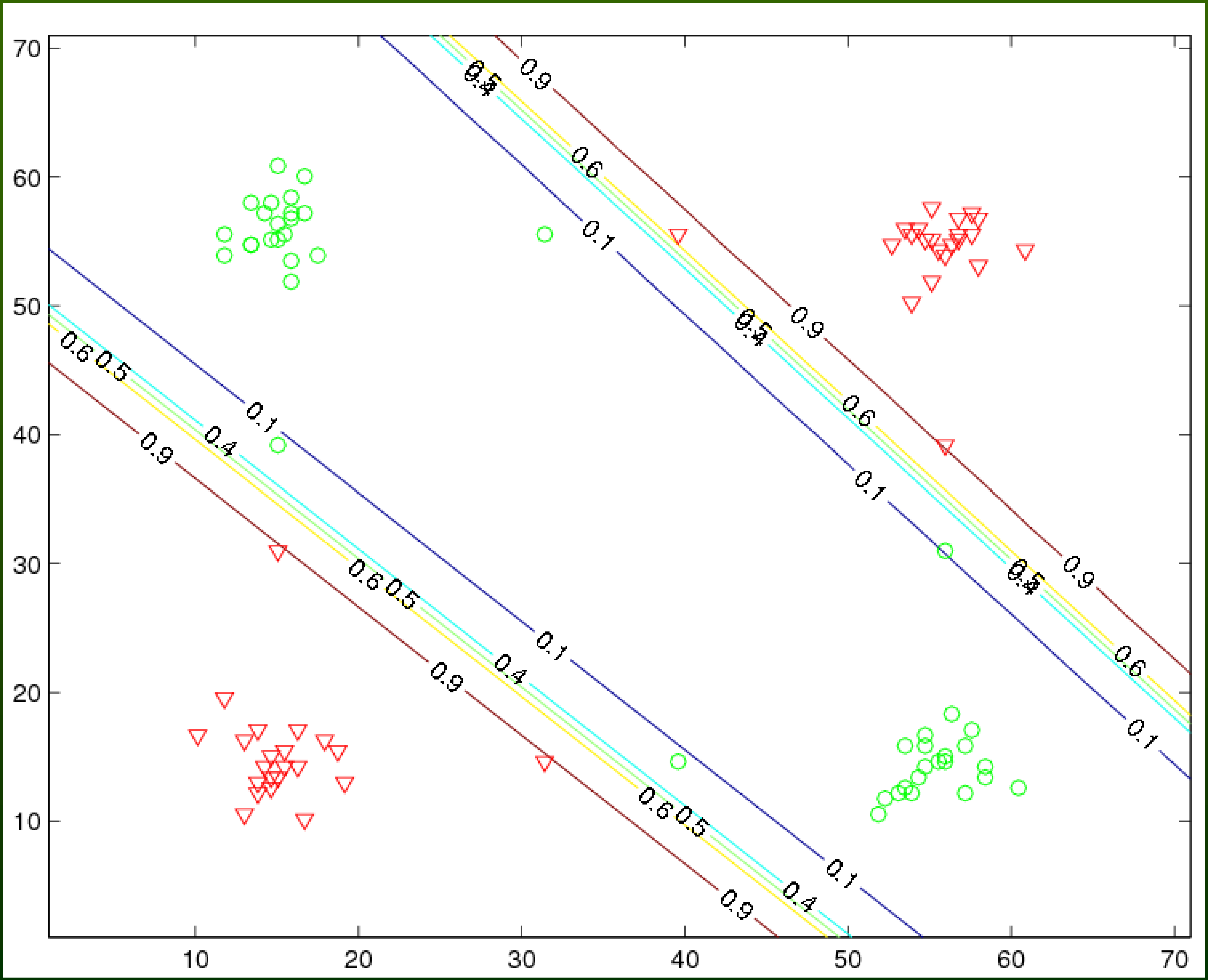
- Optimize network parameters for N_{opt} steps on SV data.
- Run feedforward step on T to determine overall accuracy and errors, take SV={X|ε(X)$\in$ [ε_{min},1$-\varepsilon_{min}$]}.
- If the accuracy increases:

  compare current network with the previous best one,
  choose the better one as the current best
- increase $\varepsilon_{min}=\varepsilon_{min}+\Delta\varepsilon$ and make forward step selecting SVs
- If the number of support vectors |SV| increases:

  decrease $\varepsilon_{min}=\varepsilon_{min}-\Delta\varepsilon$;

  decrease $\Delta\varepsilon = \Delta\varepsilon/1.2$ to avoid large changes

XOR solution



Hypothyroid data example

2 years real medical screening tests for thyroid diseases, 3772 cases with 93 primary hypothyroid and 191 compensated hypothyroid, the remaining 3488 cases are healthy; 3428 test, similar class distribution.

21 attributes (15 binary, 6 continuous), but only 2 binary attributes (on thyroxine, and thyroid surgery) are useful, therefore 8 attributes are used.

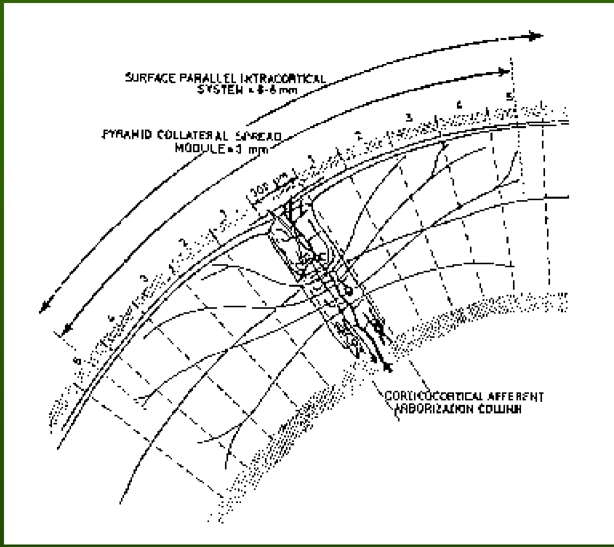
Method	% train	% test
C-MLP2LN rules	99.89	99.36
MLP+SCG, 4 neurons	99.81	99.24
SVM Minkovsky opt kernel	100.0	99.18
MLP+SCG, 4 neur, 67 SV	99.95	99.0
MLP+SCG, 12 neur.	100.0	98.8
Cascade correlation	100.0	98.5
MLP+backprop	99.60	98.5
SVM Gaussian kernel	99.76	98.4

Biological inspirations

Cortical columns may learn to respond to stimuli with complex logic, resonating in different way.

Liquid state machine (LSM; Maas, Markram 2004) – large spiking recurrent neural network, randomly connected.

S(t) => LSM (x,t), spatio-temporal pattern of activations, creating separable high dimensional projections, perceptrons can handle that.


SURFACE PARALLEL INTRACORTICAL SYSTEM ≈ 5-6 mm
PYRAMID COLLATERAL SPREAD MODULE ≈ 3 mm
300 μm
CORTICOCORTICAL AFFERENT ARBORIZATION COLUMN

Simplifications for static data:

1) Oscillators based on combination of two neurons $\sigma(W \cdot X-b) - \sigma(W \cdot X-b')$ give localized projections ⇔ specific resonant states!
Used in MLP2LN architecture for extraction of logical rules from data.

2) Single hidden layer constructive network based on random projections.

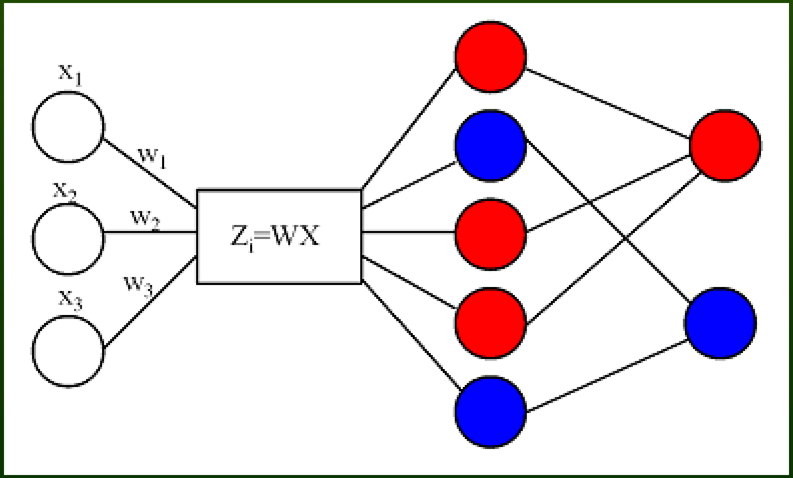
aRPM

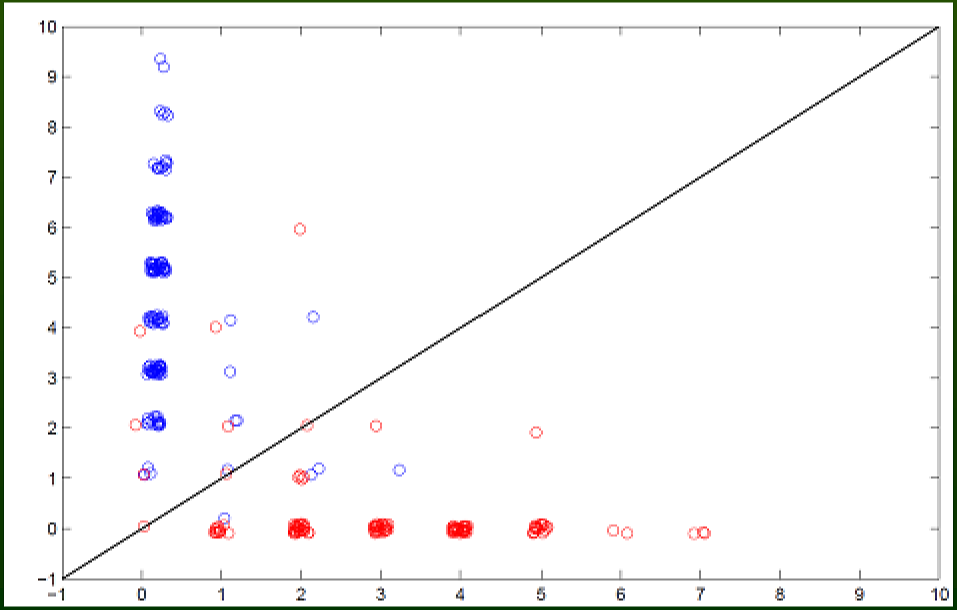
aRMP, Almost Random Projection Machine (with Hebbian learning):

generate random combinations of inputs (line projection) $z(X)=W\cdot X$,

find and isolate pure cluster $h(X)=G(z(X))$; localized kernel on projections,
estimate relevance of $h(X)$, ex. $MI(h(X),C)$,
leave only good nodes and continue until each vector activates minimum k hidden nodes.

Count how many nodes vote for each class and plot: no LDA needed!
No need for learning at all!





Main idea

Following biological inspirations - aRPM, single hidden layer constructive network based on random projections added only if useful;

easily solve highly-non-separable problems.

Kernel-based features extend the hypothesis space.
Cover theorem guarantees better separability.

Linear kernels define new features $t(X;W)=K_L(X,W)=X\cdot W$ based on a projection on the *W* direction.

Gaussian kernels $g(X,W)=\exp(-||X-W||/(2\sigma^2))$ evaluate similarity between two vectors using weighted radial distance function.

The final discriminant function is constructed as a linear combination of such kernels.

Focus on margin maximization in aRPM; new projections added only if they increase correct classification probability of those examples that are wrongly classified or are close to the decision border.

aRPM

Some projections may not be very useful, but the distribution of the training data along direction may have a range of values that includes a pure cluster of projected patterns.
This creates binary features $b_i(X) = t(X;W_i,[t_a,t_b]) \in \{0,1\}$, based on linear restricted projection in the direction W.

Good candidate feature should cover some minimal number η of training vectors.

Features based on kernels - here only Gaussian kernels with several values of dispersion σ $g(X;X_i,\sigma)=\exp(-||X_i-X||^2/2\sigma^2)$.

Local kernel features have values close to zero except around their support vectors X_i.

Their usefulness is limited to the neighborhood $O(X_i)$ in which $g_i(X)>\epsilon$.

aRPM with margin optimization

In the WTA $|A(C|X)-A(\neg C|X)|$ estimates distance from the decision border.

Specifying confidence of the model for vector $X \in C$ using logistic function:

$F(X)=1/(1+\exp(-(A(C|X)-A(\neg C|X))))$

gives values around 1 if X is on the correct side and far from the border, and goes to zero if it is on the wrong side.

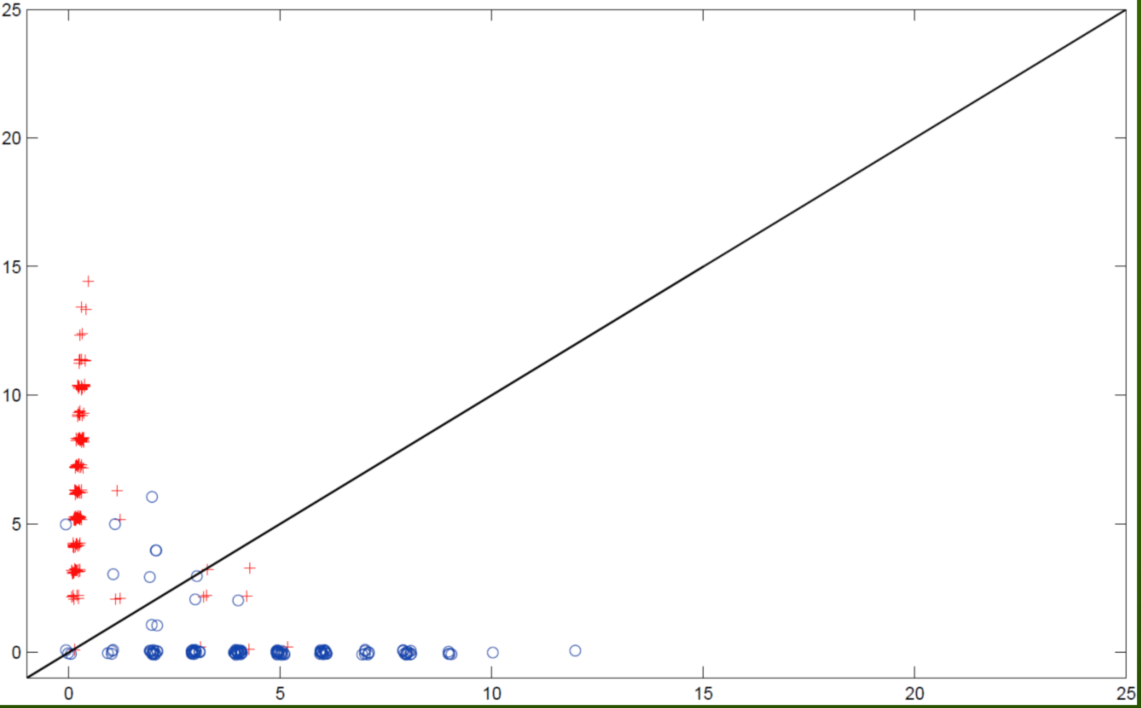
Total confidence in the model may then be estimated by summing over all vectors.

The final effect of adding new feature $h(X)$ to the total confidence measure is therefore:

$U(H,h)=\Sigma\ (F(X;H+h) - F(X;H))$

If $U(H,h)>\alpha$ than the new feature is accepted providing a larger margin.

To make final decision aRPM with margin maximization uses WTA mechanism or LD.

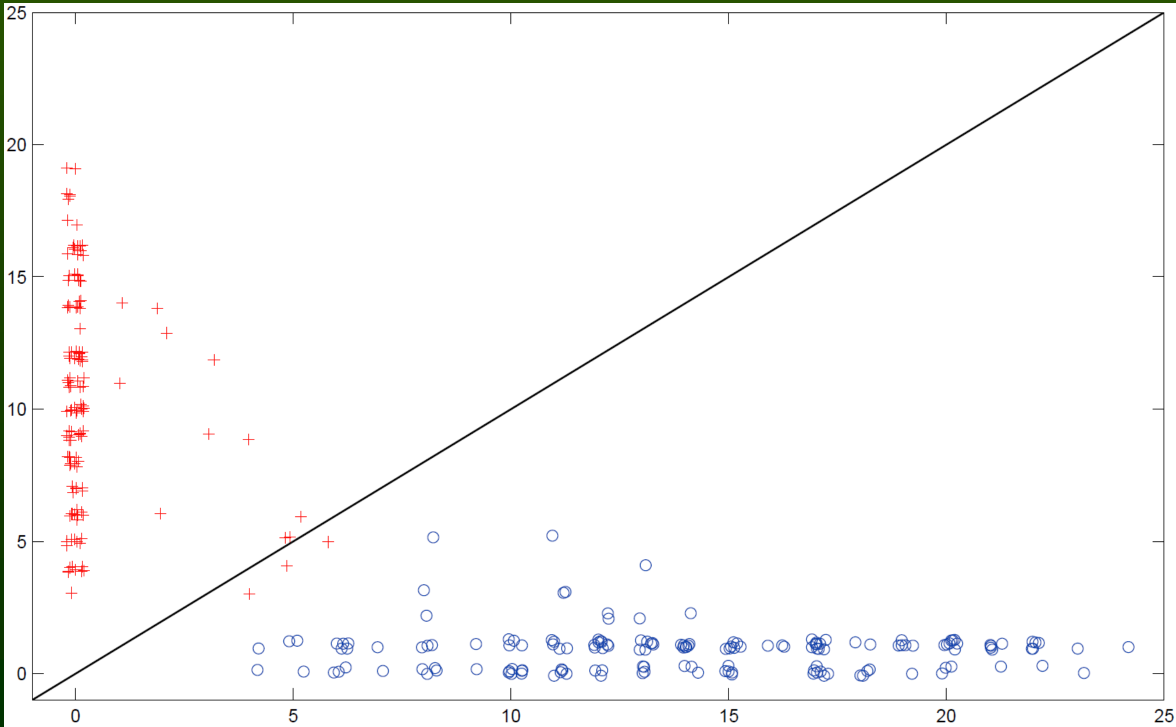


Heart Statlog data

no margin maximization

Axes: number of voting nodes for each vector; color = class

with margin maximization

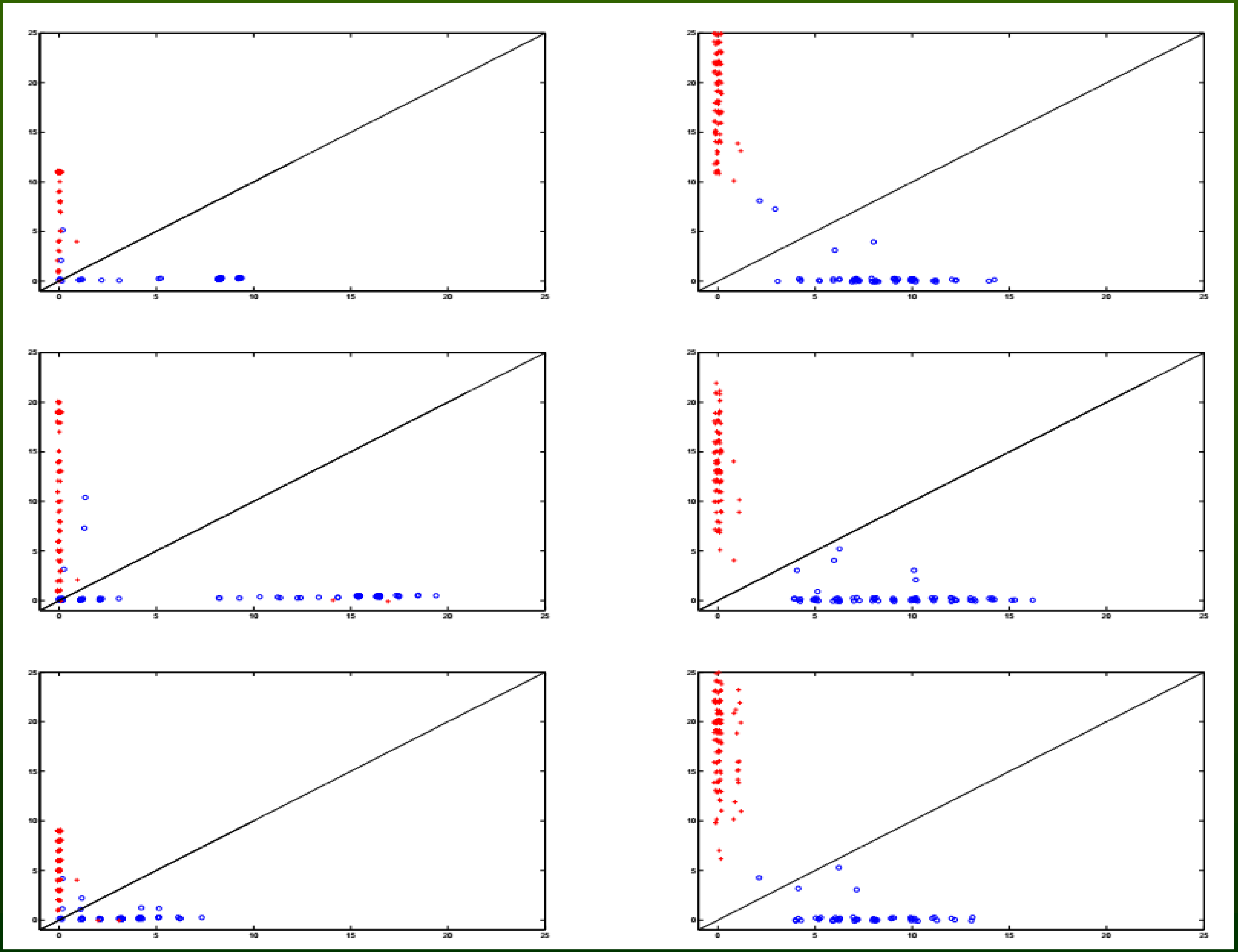


More vectors activate many nodes from correct class, few vectors are close to the decision border (majority voting).

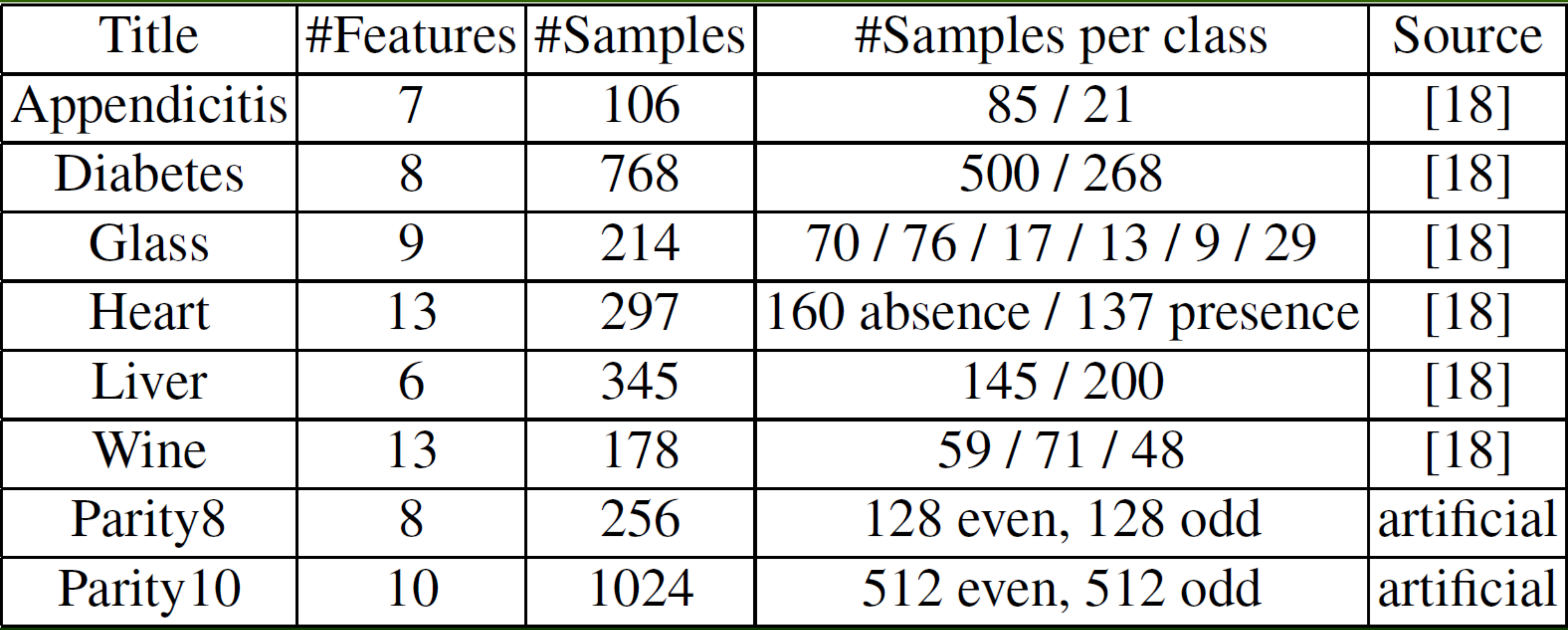
no margin maximization

with margin maximization

Wine data, 3 classes, pairwise

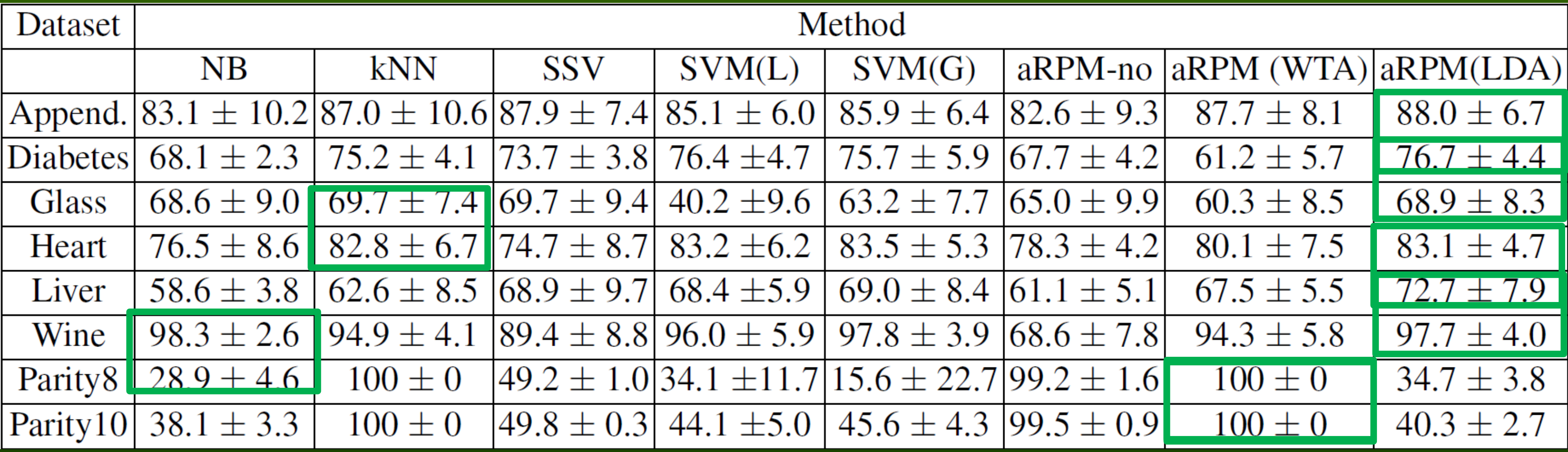


Datasets



Title	#Features	#Samples	#Samples per class	Source
Appendicitis	7	106	85 / 21	[18]
Diabetes	8	768	500 / 268	[18]
Glass	9	214	70 / 76 / 17 / 13 / 9 / 29	[18]
Heart	13	297	160 absence / 137 presence	[18]
Liver	6	345	145 / 200	[18]
Wine	13	178	59 / 71 / 48	[18]
Parity8	8	256	128 even, 128 odd	artificial
Parity10	10	1024	512 even, 512 odd	artificial

aRMP results



Dataset	Method							
	NB	kNN	SSV	SVM(L)	SVM(G)	aRPM-no	aRPM (WTA)	aRPM(LDA)
Append.	83.1 ± 10.2	87.0 ± 10.6	87.9 ± 7.4	85.1 ± 6.0	85.9 ± 6.4	82.6 ± 9.3	87.7 ± 8.1	$\boxed{88.0 \pm 6.7}$
Diabetes	68.1 ± 2.3	75.2 ± 4.1	73.7 ± 3.8	76.4 ±4.7	75.7 ± 5.9	67.7 ± 4.2	61.2 ± 5.7	$\boxed{76.7 \pm 4.4}$
Glass	68.6 ± 9.0	$\boxed{69.7 \pm 7.4}$	69.7 ± 9.4	40.2 ±9.6	63.2 ± 7.7	65.0 ± 9.9	60.3 ± 8.5	$\boxed{68.9 \pm 8.3}$
Heart	76.5 ± 8.6	$\boxed{82.8 \pm 6.7}$	74.7 ± 8.7	83.2 ±6.2	83.5 ± 5.3	78.3 ± 4.2	80.1 ± 7.5	$\boxed{83.1 \pm 4.7}$
Liver	58.6 ± 3.8	62.6 ± 8.5	68.9 ± 9.7	68.4 ±5.9	69.0 ± 8.4	61.1 ± 5.1	67.5 ± 5.5	$\boxed{72.7 \pm 7.9}$
Wine	$\boxed{98.3 \pm 2.6}$	94.9 ± 4.1	89.4 ± 8.8	96.0 ± 5.9	97.8 ± 3.9	68.6 ± 7.8	94.3 ± 5.8	$\boxed{97.7 \pm 4.0}$
Parity8	$\boxed{28.9 \pm 4.6}$	100 ± 0	49.2 ± 1.0	34.1 ±11.7	15.6 ± 22.7	99.2 ± 1.6	$\boxed{100 \pm 0}$	34.7 ± 3.8
Parity10	38.1 ± 3.3	100 ± 0	49.8 ± 0.3	44.1 ±5.0	45.6 ± 4.3	99.5 ± 0.9	$\boxed{100 \pm 0}$	40.3 ± 2.7

Simplest method that solves highly-non separable problems like parity!

aRPM with locally optimized kernels

To create multi-resolution kernel features – first with large σ (smooth decision borders), then smaller σ (features more localized).

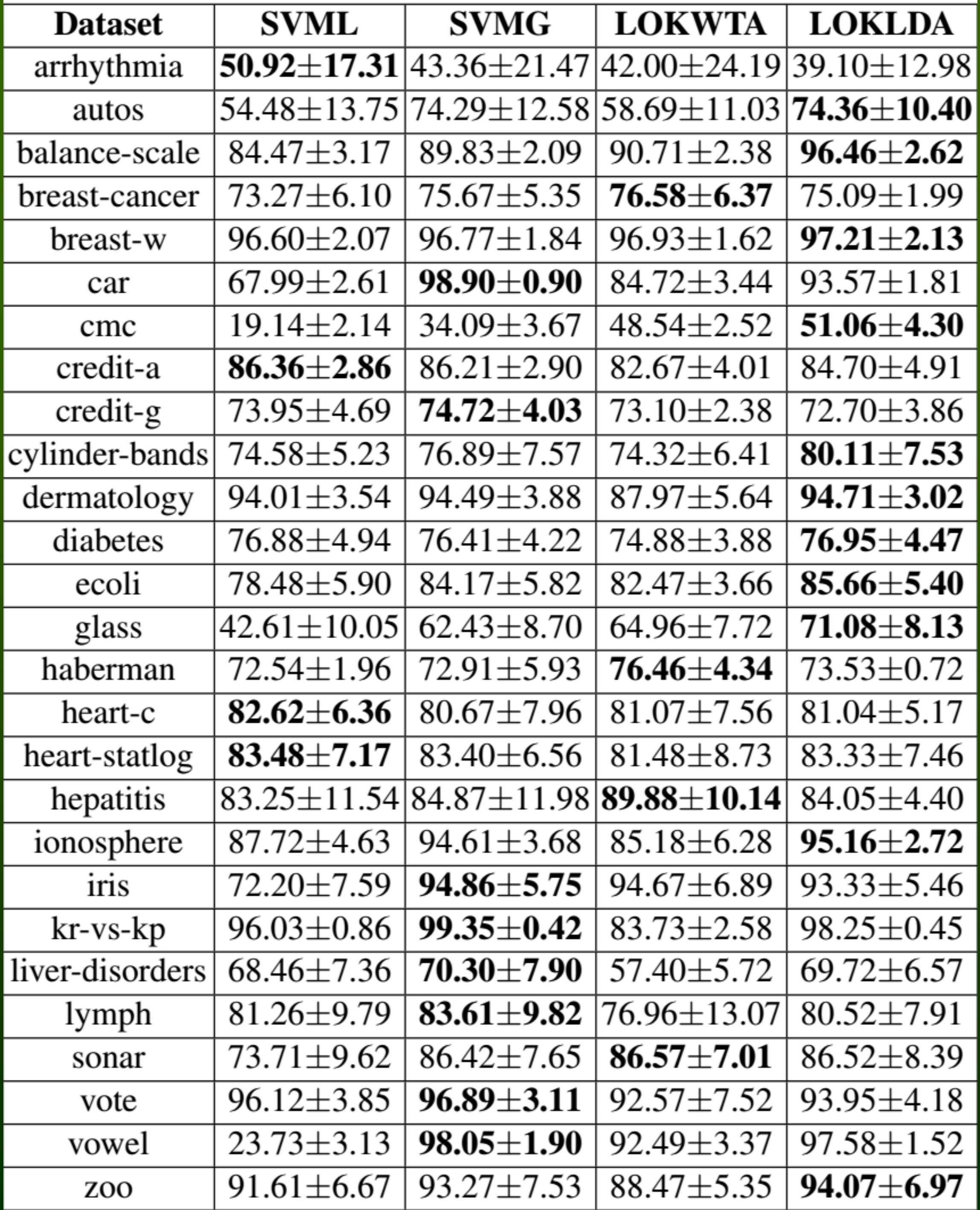
The candidate feature is converted into a permanent node only if it increases classification margin.

Incremental algorithm, expand feature space until no improvements; move vectors away from decision border.

WTA - sum of the activation of hidden nodes. Projections with added intervals give binary activations $b_i(X)$, but the values of kernel features $g(X;X_i,\sigma)$ are summed, giving a total activation $A(C|X)$ for each class.

Plotting $A(C|X)$ versus $A(\neg C|X)$ for each vector leads to scatterograms, giving an idea how far is a given vector from the decision border.

aRMP with LOK results, simplest version



Dataset	SVML	SVMG	LOKWTA	LOKLDA
arrhythmia	**50.92±17.31**	43.36±21.47	42.00±24.19	39.10±12.98
autos	54.48±13.75	74.29±12.58	58.69±11.03	**74.36±10.40**
balance-scale	84.47±3.17	89.83±2.09	90.71±2.38	**96.46±2.62**
breast-cancer	73.27±6.10	75.67±5.35	**76.58±6.37**	75.09±1.99
breast-w	96.60±2.07	96.77±1.84	96.93±1.62	**97.21±2.13**
car	67.99±2.61	**98.90±0.90**	84.72±3.44	93.57±1.81
cmc	19.14±2.14	34.09±3.67	48.54±2.52	**51.06±4.30**
credit-a	**86.36±2.86**	86.21±2.90	82.67±4.01	84.70±4.91
credit-g	73.95±4.69	**74.72±4.03**	73.10±2.38	72.70±3.86
cylinder-bands	74.58±5.23	76.89±7.57	74.32±6.41	**80.11±7.53**
dermatology	94.01±3.54	94.49±3.88	87.97±5.64	**94.71±3.02**
diabetes	76.88±4.94	76.41±4.22	74.88±3.88	**76.95±4.47**
ecoli	78.48±5.90	84.17±5.82	82.47±3.66	**85.66±5.40**
glass	42.61±10.05	62.43±8.70	64.96±7.72	**71.08±8.13**
haberman	72.54±1.96	72.91±5.93	**76.46±4.34**	73.53±0.72
heart-c	**82.62±6.36**	80.67±7.96	81.07±7.56	81.04±5.17
heart-statlog	**83.48±7.17**	83.40±6.56	81.48±8.73	83.33±7.46
hepatitis	83.25±11.54	84.87±11.98	**89.88±10.14**	84.05±4.40
ionosphere	87.72±4.63	94.61±3.68	85.18±6.28	**95.16±2.72**
iris	72.20±7.59	**94.86±5.75**	94.67±6.89	93.33±5.46
kr-vs-kp	96.03±0.86	**99.35±0.42**	83.73±2.58	98.25±0.45
liver-disorders	68.46±7.36	**70.30±7.90**	57.40±5.72	69.72±6.57
lymph	81.26±9.79	**83.61±9.82**	76.96±13.07	80.52±7.91
sonar	73.71±9.62	86.42±7.65	**86.57±7.01**	86.52±8.39
vote	96.12±3.85	**96.89±3.11**	92.57±7.52	93.95±4.18
vowel	23.73±3.13	**98.05±1.90**	92.49±3.37	97.58±1.52
zoo	91.61±6.67	93.27±7.53	88.47±5.35	**94.07±6.97**

aRMP conclusions

aRPM has many advantages that has not been yet explored.
Some work on ELM (Extreme Learning Machines) goes in this direction.

- Biological plausibility – virtually no learning involved, just generation and selection of features.
- Solves fast learning problem. Focus on generation of new features followed by the WTA, simpler and faster than typical NN networks.
- Selection of network nodes to ensure wide margins.
- Adding kernel features, locally optimized, relations to kernel SVM
- Feature selection and construction, finding interesting views on the data is the basis of natural categorization and learning processes.
- Scatterograms of WTA output show effects of margin optimization, and allow for estimation of confidence in classification of a given data.

Further improvements: admission of impure clusters instead of binary features, use of other kernel features, locally optimized kernels, selection of candidate SV, optimization of the algorithm.

Foundations of Machine Learning

Principles: information compression

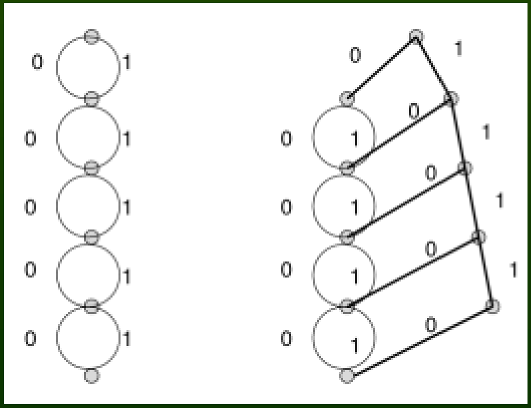
Neural information processing in perception and cognition: information compression, or algorithmic complexity.
In computing: minimum length (message, description) encoding.

Wolff (2006): all cognition and computation is information compression!
Analysis and production of natural language, fuzzy pattern recognition, probabilistic reasoning and unsupervised inductive learning.

Talks about multiple alignment, unification and search, but only models for sequential data and 1D alignment have been demonstrated.

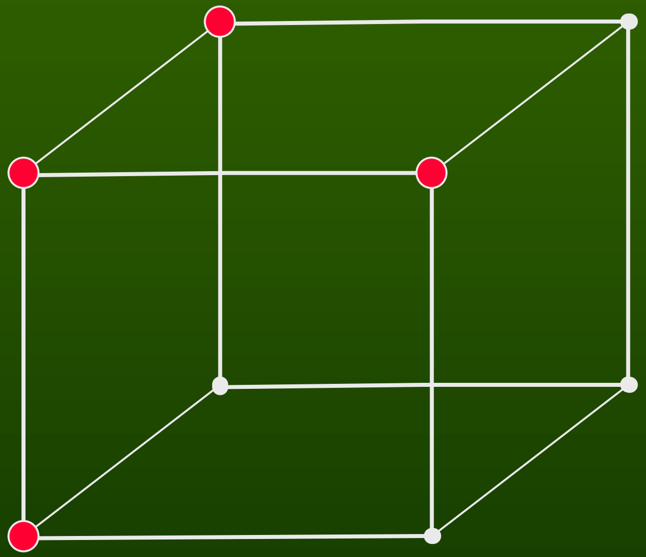
Information compression: encoding new information in terms of old.

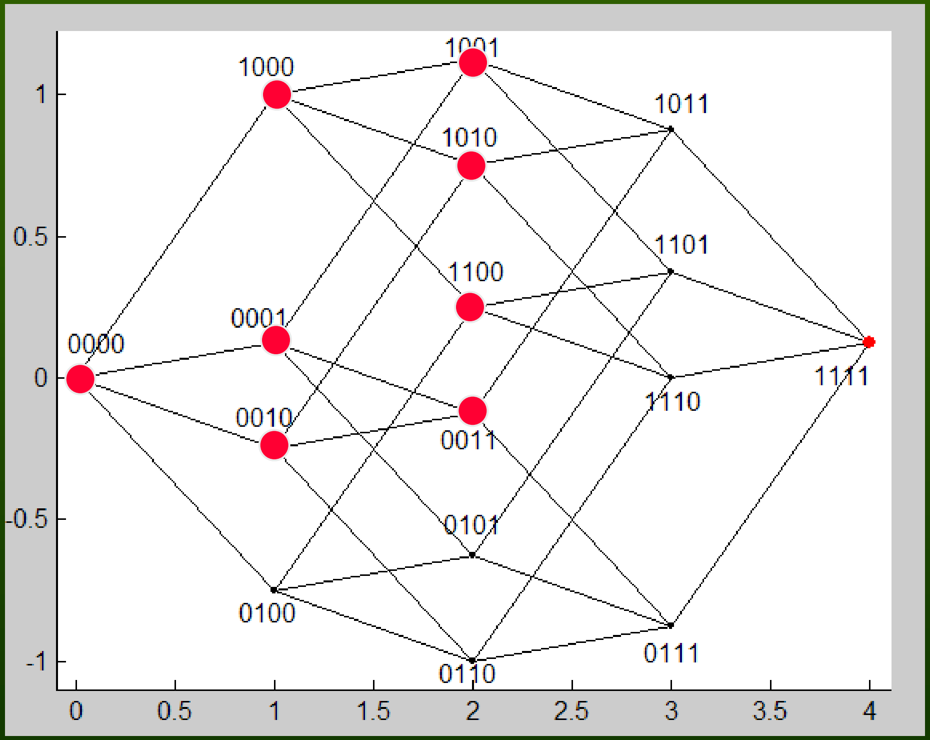
Measure of syntactic and semantic information (Duch, Jankowski 1994); based on the size of the minimal graph representing a given data structure or knowledge-base specification, thus it goes beyond alignment;
real information = what model cannot predict.
"Surprise" and curiosity measures: Pfaffelhuber (1972), Palm, Schmidhuber, Baldi … all based on the same idea.



What can be learned?

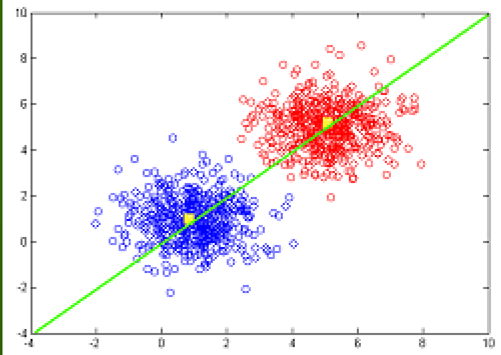
Linearly separable or almost separable problems are relatively simple – deform planes or add dimensions to make data separable.

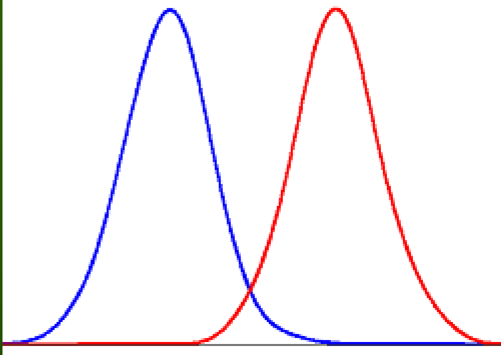




How to define "slightly non-separable", or relatively easy to learn?
Now we have only separable problems and one vast realm of the rest.

Easy problems





- Approximately linearly separable problems in the original feature space: linear discrimination is sufficient (always worth trying, but no-one does!).
- Simple topological deformation of decision borders is sufficient – linear separation is then possible in extended/transformed spaces.

This is frequently sufficient for pattern recognition problems (more than half of UCI problems).

- RBF/MLP networks with one hidden layer also solve such problems easily, but convergence/generalization for anything more complex than XOR is problematic.

SVM adds new features to “flatten” the decision border:

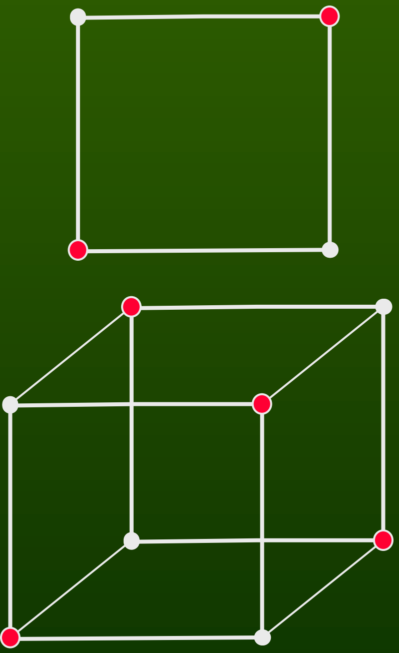
$$\mathbf{X} = (x_1, x_2, \ldots x_n); \quad z_i(\mathbf{X}) = K\left(\mathbf{X}^{(i)}, \mathbf{X}\right)$$

achieving larger margins/separability in the X+Z space.

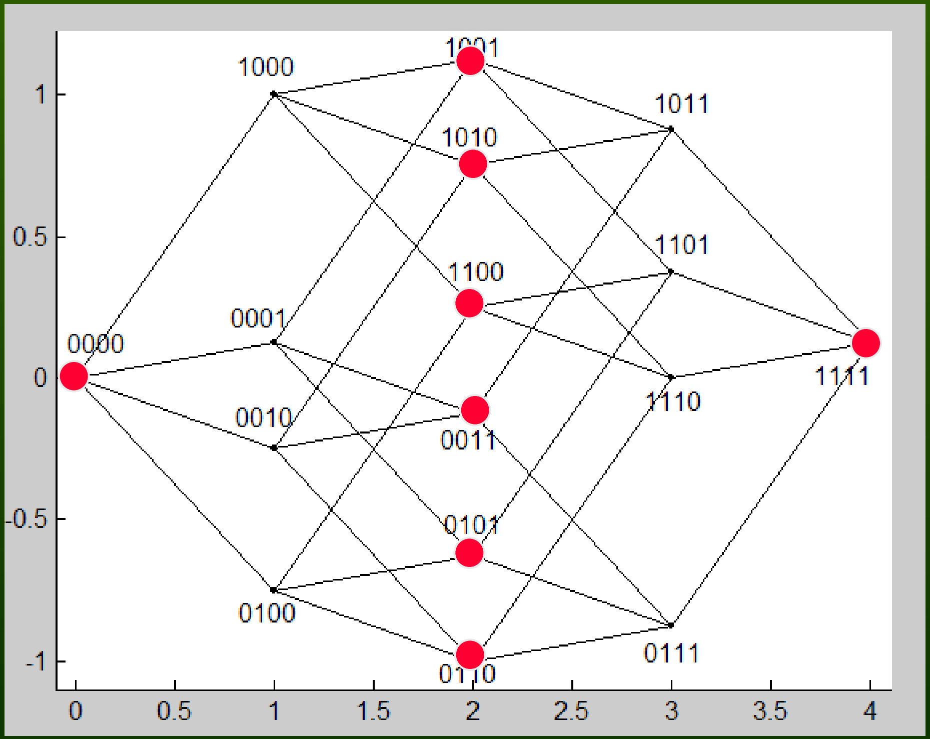
Neurons learning complex logic

Boole'an functions are difficult to learn, n bits but 2^n nodes => combinatorial complexity; similarity is not useful, for parity all neighbors are from the wrong class. MLP networks have difficulty to learn functions that are highly non-separable.

Ex. of 2-4D parity problems.

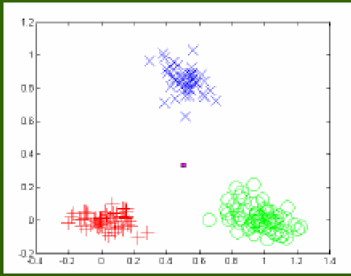


Neural logic can solve it without counting; find a good point of view.



Projection on W=(111 ... 111) gives clusters with 0, 1, 2 ... n bits; solution requires abstract imagination + easy categorization.

Easy and difficult problems



Linear separation: good goal if simple topological deformation of decision borders is sufficient.
RBF/MLP networks with one hidden layer solve such problems.

Difficult problems: disjoint clusters, combinatorial problems, complex logic.
Networks with localized functions need exponentially large number of nodes.

Q1: How to characterize complexity of Boolean functions? Non-separability is not sufficient.
Q2: What is the simplest model for a given type of data?
Q3: How to learn such model in an automatic way?

Boolean functions: n=2 vectors V={00,01,10,11},
Boolean functions {0000, 0001 ... 1111}, ex. 0001 = AND, 0110 = OR,
each function is identified by number from 0 to 15 = 2^K-1.

For *n* bits there are $K=2^n$ binary vectors that can be represented as vertices of *n*-dimensional hypercube.
Each Boolean function is identified by K bits.
BoolF(B_i) = 0 or 1 for i=1..K, leads to the 2^K Boolean functions.

Boolean functions

n=2, 16 functions, 12 separable, 4 not separable.

n=3, 256 f, 104 separable (41%), 152 not separable.

n=4, 64K=65536, only 1880 separable (3%)

n=5, 4G, but << 1% separable ... bad news!

Most bioinformatics or neuroimaging data may require n >100.

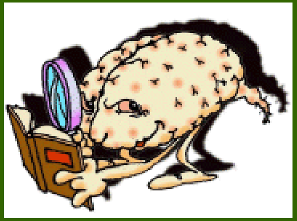
Existing methods may learn some non-separable functions,
but most functions cannot be learned !

Example: *n*-bit parity problem; many papers in top journals.
No off-the-shelf systems are able to solve such problems.

For all parity problems SVM is below base rate!
Such problems are solved only by special neural architectures or special classifiers – if the type of function is known.

But parity is still trivial ... solved by $y = \cos\left(\omega \sum_{i=1}^{n} b_i\right)$

Goal of learning



If simple topological deformation of decision borders is sufficient linear separation is achieved in high dimensional spaces, “flattening” non-linear decision borders; this is frequently the case in pattern recognition problems. RBF/MLP networks with one hidden layer solve the problem.

For complex logic this is not sufficient; networks with localized functions need exponentially large number of nodes.

Such situations arise in AI reasoning problems, real perception, object recognition, text analysis, bioinformatics ...

Linear separation is too difficult, set an easier goal.
Linear separation: projection on 2 half-lines in the kernel space:
line $y=WX$, with $y<0$ for class – and $y>0$ for class +.

Simplest extension: **separation into k-intervals, or k-separability**.
For parity: find direction W with minimum # of intervals, $y=W{\cdot}X$

k-sep learning

Try to find lowest k with good solution:

- Assume k=2 (linear separability), try to find a good solution;
  MSE error criterion
- $$E(\mathbf{W},\theta)=\sum_{\mathbf{X}}\left(y(\mathbf{X};\mathbf{W})-C(\mathbf{X})\right)^2$$

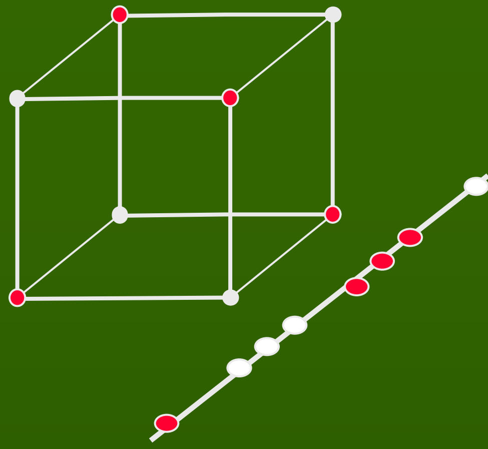
- if k=2 is not sufficient, try k=3; two possibilities are C_+,C_-,C_+ and C_-, C_+, C_- this requires only one interval for the middle class;
- if k<4 is not sufficient, try k=4; two possibilities are C_+, C_-, C_+, C_- and C_-, C_+, C_-, C_+ this requires one closed and one open interval.

Network solution ⇔ to minimization of specific cost function.

$$E(\mathbf{W},\lambda_1,\lambda_2)=\sum_{\mathbf{X}}\left(y(\mathbf{X};\mathbf{W})-C(\mathbf{X})\right)^2+\lambda_1\sum_{\mathbf{X}}\left(1-C(\mathbf{X})\right)y(\mathbf{X};\mathbf{W})$$
$$-\lambda_2\sum_{\mathbf{X}}C(\mathbf{X})y(\mathbf{X};\mathbf{W})$$

First term = MSE, second penalty for “impure” clusters, third term = reward for the large clusters.

3D case



3-bit functions: $X=[b_1b_2b_3]$, from [0,0,0] to [1,1,1]

$f(b_1,b_2,b_3)$ and $\neg f(b_1,b_2,b_3)$ are symmetric (color change)

8 cube vertices, $2^8=256$ Boolean functions.

0 to 8 red vertices: 1, 8, 28, 56, 70, 56, 28, 8, 1 functions.

For optimized direction **W** on all 2^8 functions index projection W·X gives:

k=1 in 2 cases, all 8 vectors in 1 cluster (all 8 black or all 8 white)

k=2 in 14 cases, 8 vectors in 2 clusters (linearly separable)

k=3 in 42 cases, clusters E O E or O E O   Even, Odd

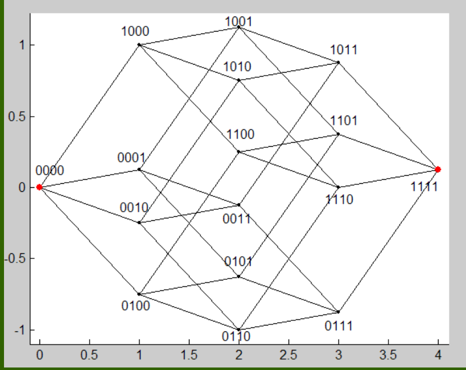
k=4 in 70 cases, clusters E O E O or O E O E

Symmetrically, k=5-8 for 70, 42, 14, 2 cases.

Most logical functions have 4 or 5-separable projections.

Learning = find best set of projections for each function.

Enforcing separability on k-separable data is hard for k>2, but assuming k-separability as a goal makes learning easier.



4D case

4-bit functions: $X=[b_1b_2b_3b_4]$, from [0,0,0,0] to [1,1,1,1]

16 cube vertices, 2^{16}=65636=64K functions.

Random initialization of a single perceptron has 39.2% chance of creating 8 or 9 clusters for the 4-bit data.

Learning optimal directions **W** finds:

k=1 in 2 cases, all 16 vectors in 1 cluster (all black or all white)

k=2 in 2.9% cases (or 1880), 16 vectors in 2 clusters (linearly sep)

k=3 in 22% of all cases, clusters B R B or W R W

k=4 in 45% of all cases, clusters R W R W or W R W R

k=5 in 29% of all cases.

Hypothesis: for n-bits highest k=n+1 ?

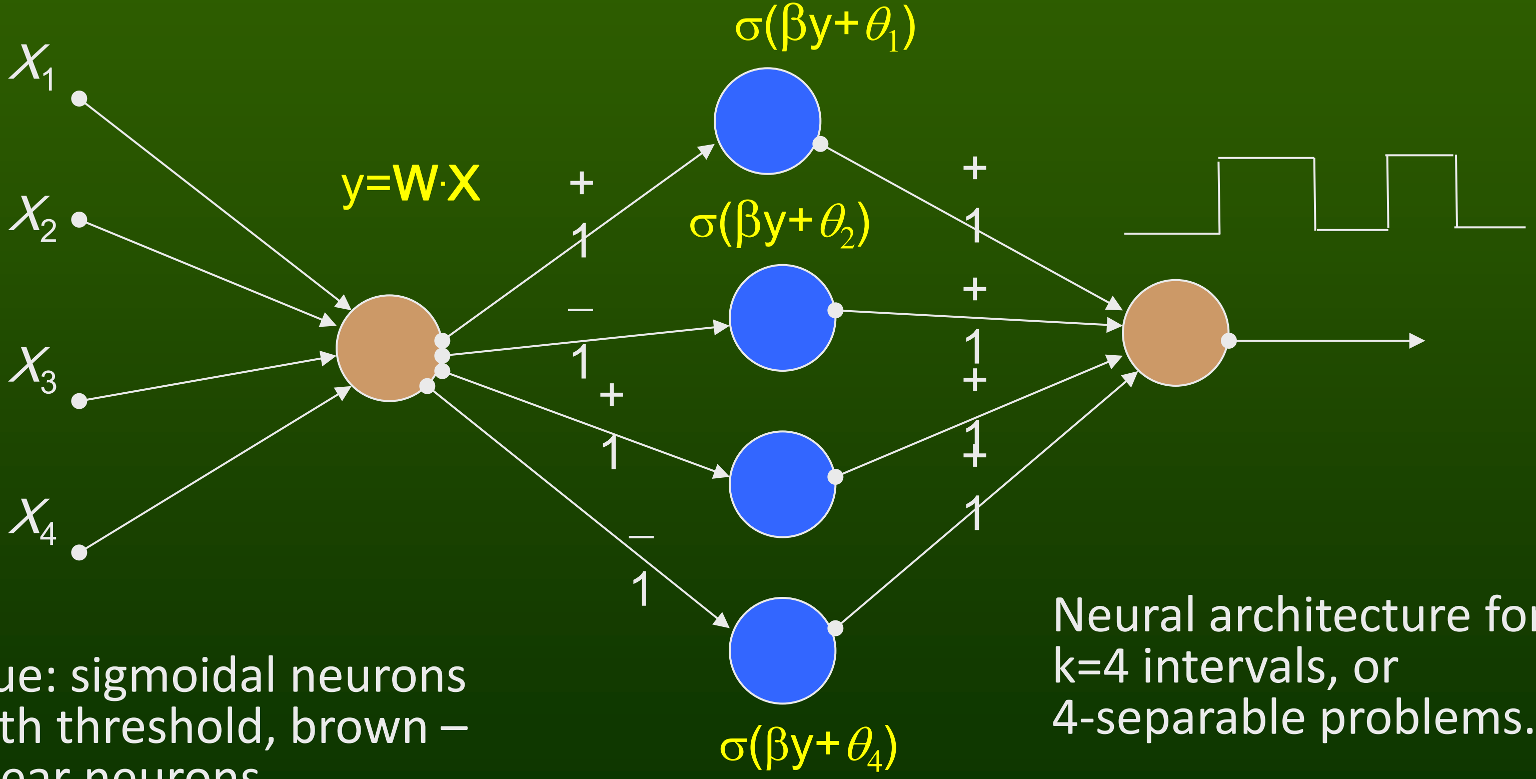
For 5-bits there are 32 vertices and already 2^{32}=4G=$4.3 \cdot 10^9$ functions.

Most are 5-separable, less than 1% is linearly separable!

Network solution

Can one learn a simplest model for highly complex logical functions?

2-separable (linearly separable) problems are easy;
non separable problems may be broken into k-separable, k>2.



X_1
X_2
X_3
X_4
y=W·X
$\sigma(\beta y+\theta_1)$
$\sigma(\beta y+\theta_2)$
$\sigma(\beta y+\theta_4)$
+1
–1
+1
–1
+1
+1
+1
+1

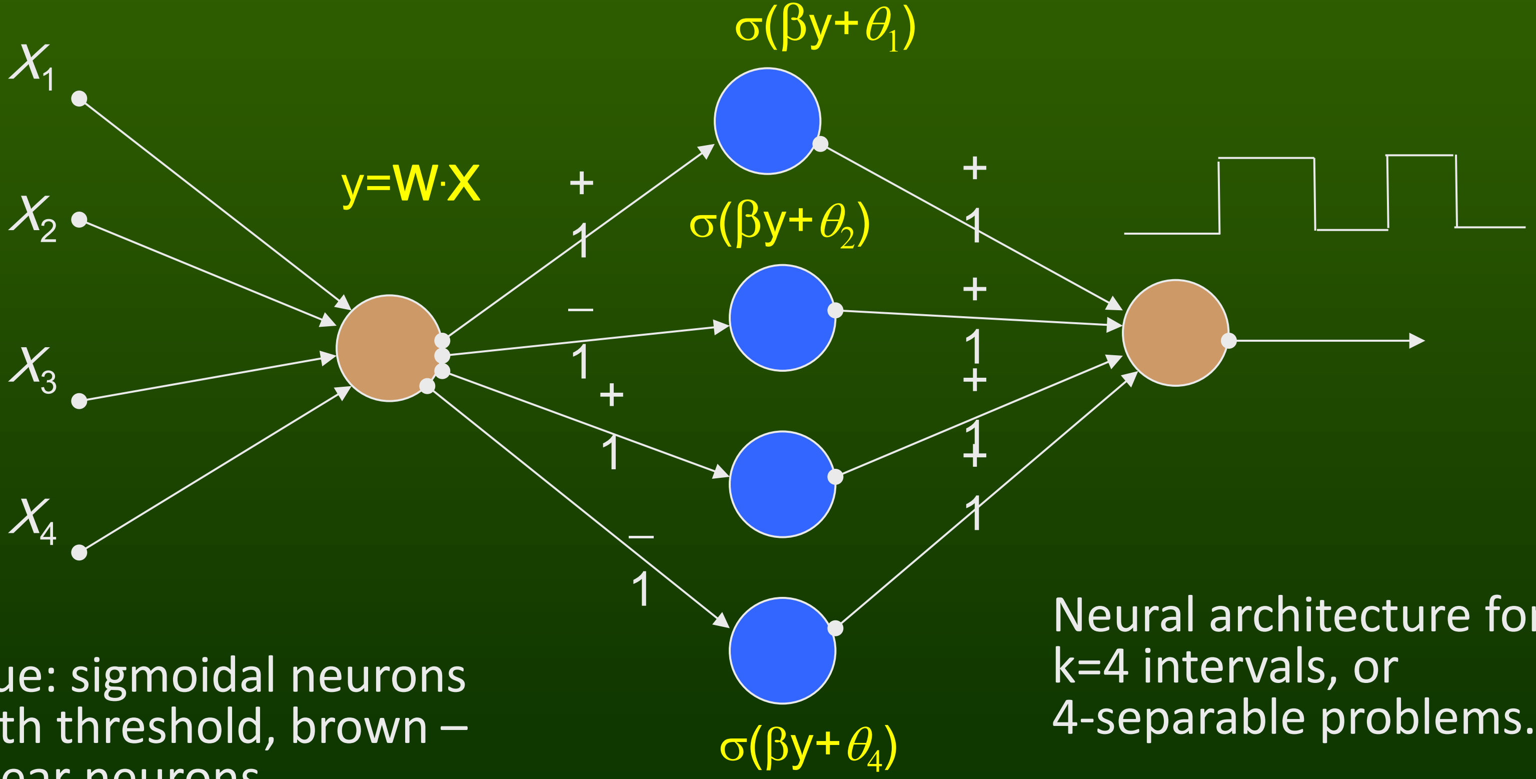
Blue: sigmoidal neurons with threshold, brown – linear neurons.

Neural architecture for k=4 intervals, or 4-separable problems.

k-separability

Can one learn all Boolean functions?

Problems may be classified as 2-separable (linear separability); non separable problems may be broken into k-separable, k>2.



X_1
X_2
X_3
X_4
$y = \mathbf{W} \cdot \mathbf{X}$
$\sigma(\beta y+\theta_1)$
$\sigma(\beta y+\theta_2)$
$\sigma(\beta y+\theta_4)$
+1
−1
+1
−1
+1
+1
+1
+1

Blue: sigmoidal neurons with threshold, brown – linear neurons.

Neural architecture for k=4 intervals, or 4-separable problems.

QPC, Projection Pursuit

What is needed to learn data with complex logic?

- cluster non-local areas in the X space, use W·X
- capture local clusters after transformation, use G(W·X-θ)

SVMs fail because the number of directions W that should be considered grows exponentially with the size of the problem *n*.

What will solve it? Projected clusters!

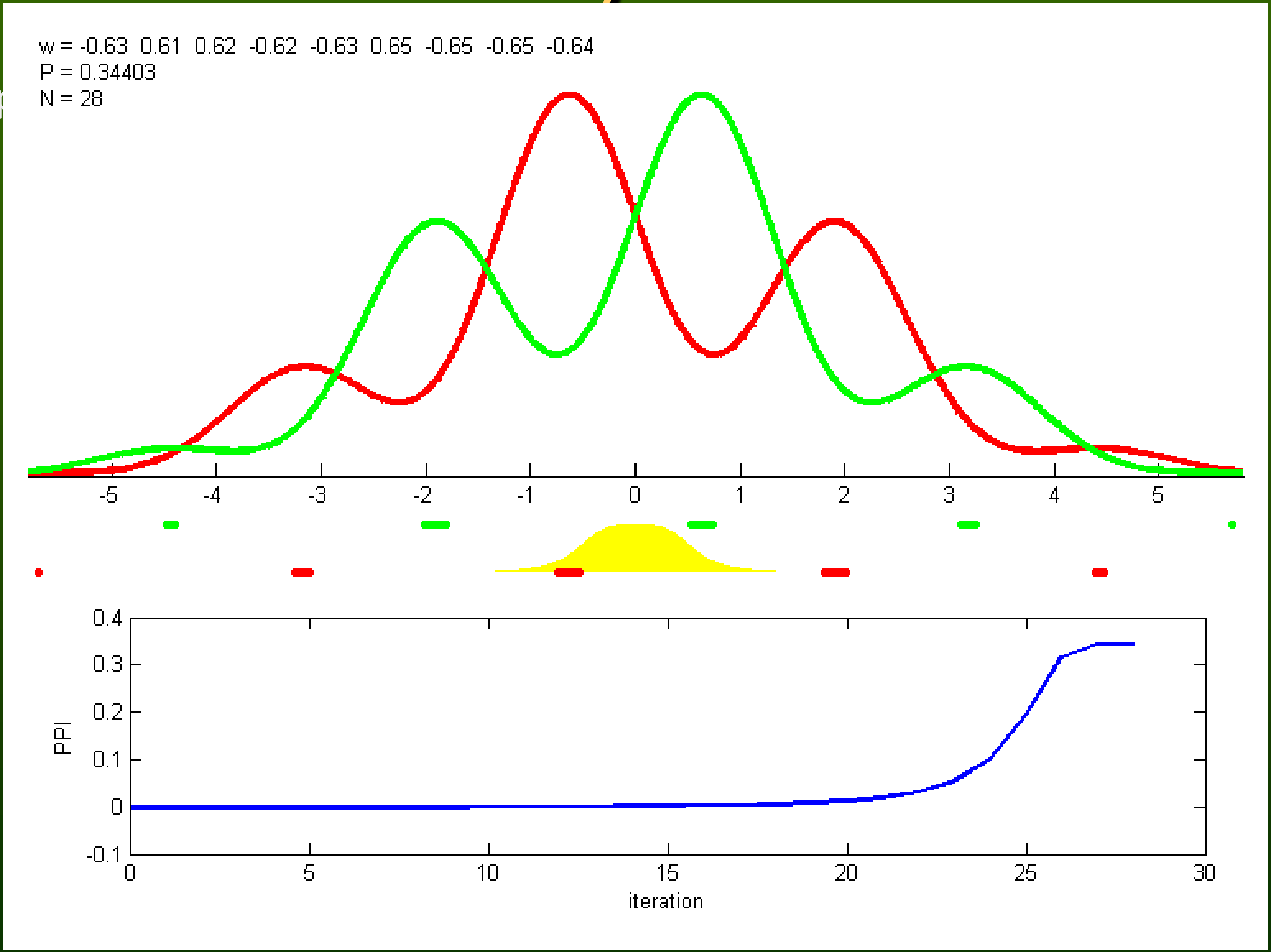
1. A class of constructive neural network solution with G(W·X-θ) functions combining non-local/local projections, with special training algorithms.
2. Maximize the leave-one-out error after projection: take some localized function *G*, count in a soft way cases from the same class as X_k.

$$Q(\mathbf{W}) = \sum_{\mathbf{X}} \left[A^{+} \sum_{\mathbf{X}_k \in C} G\big(\mathbf{W}\cdot(\mathbf{X}-\mathbf{X}_k)\big) - A^{-} \sum_{\mathbf{X}_k \notin C} G\big(\mathbf{W}\cdot(\mathbf{X}-\mathbf{X}_k)\big) \right]$$

Grouping and separation; projection may be done directly to 1 or 2D for visualization, or higher D for dimensionality reduction, if W has *d* columns.

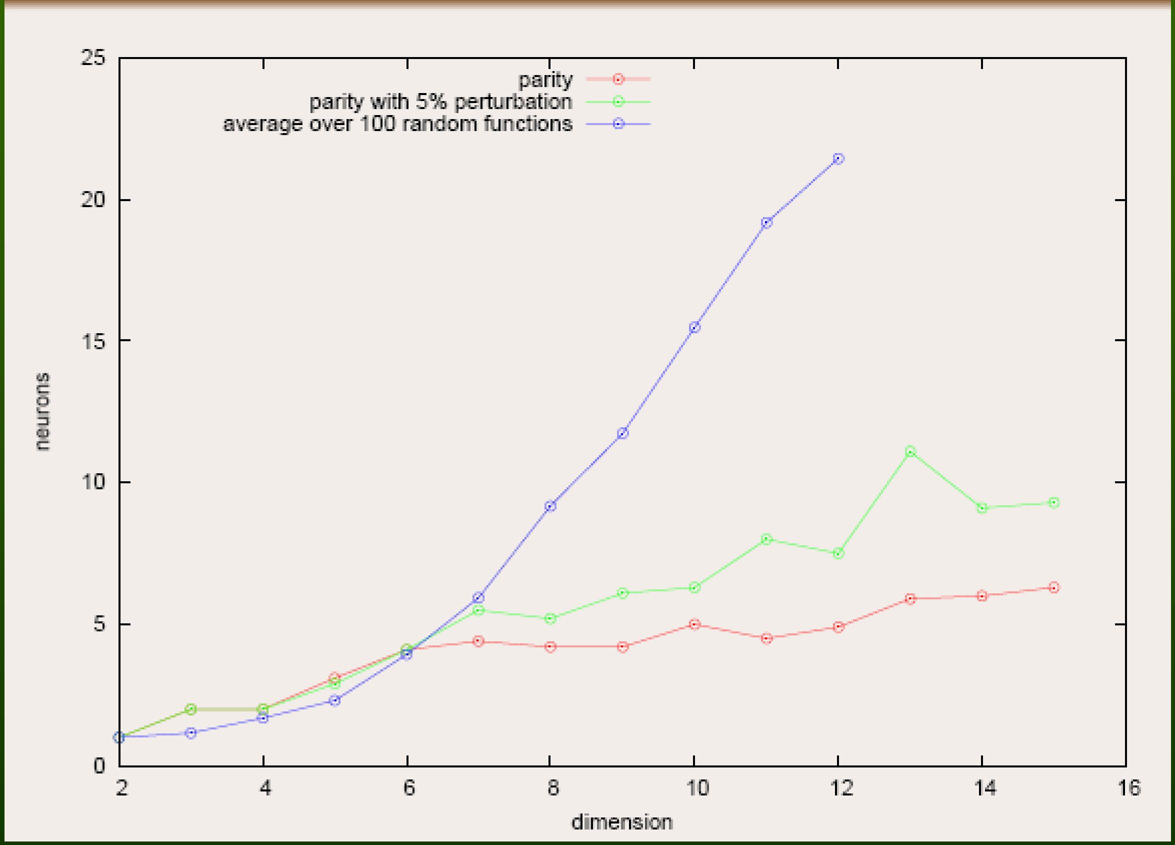
Parity n=9

Simp



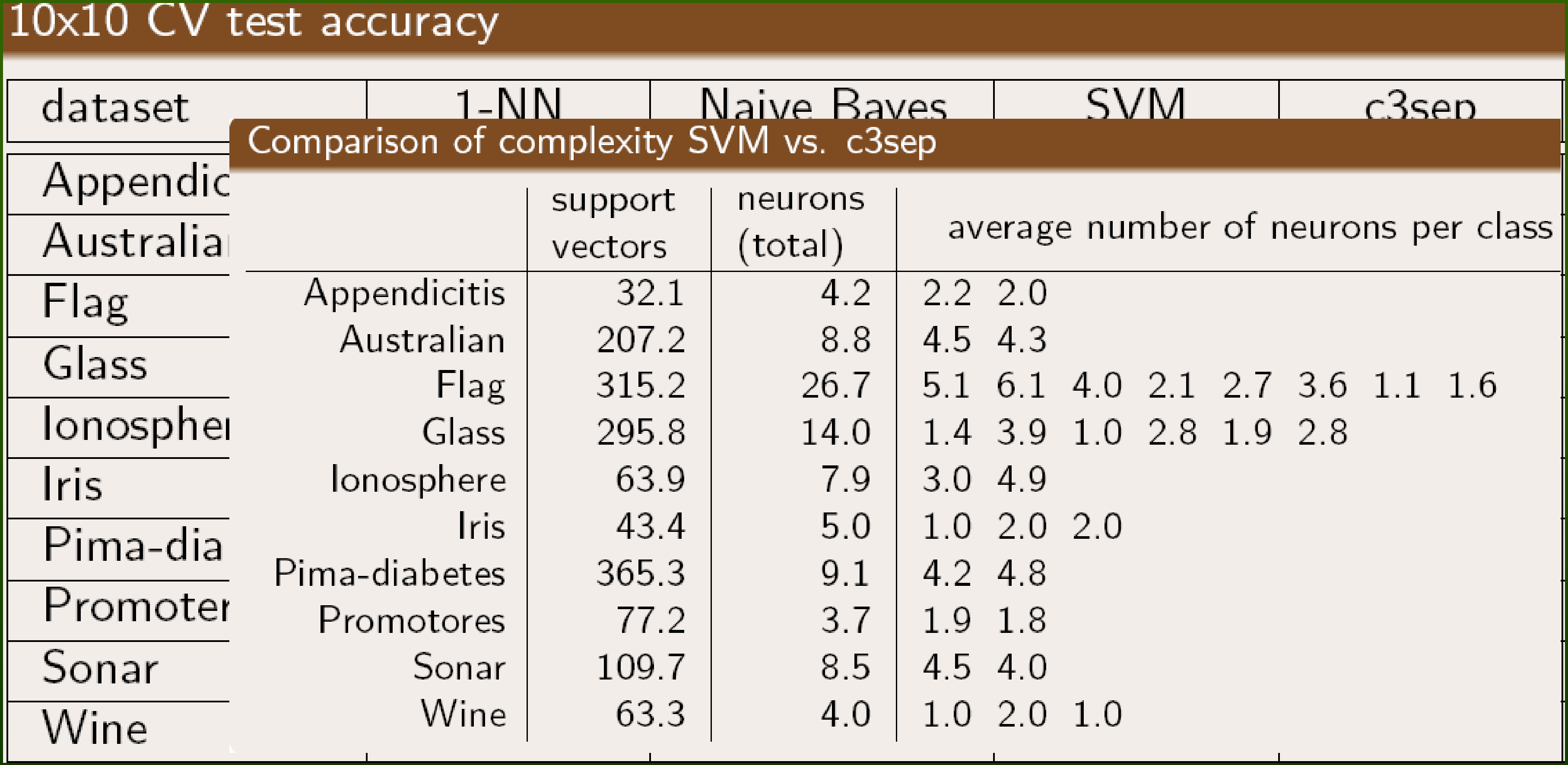
Learning hard functions

Average no. of neurons



Training almost perfect for parity, with linear growth in the number of vectors for k-sep. solution created by the constructive neural algorithm.

Real data



10x10 CV test accuracy

dataset	1-NN	Naive Bayes	SVM	c3sep
Appendic				
Australia				
Flag				
Glass				
Ionospher				
Iris				
Pima-dia				
Promoter				
Sonar				
Wine				

Comparison of complexity SVM vs. c3sep

	support vectors	neurons (total)	average number of neurons per class							
Appendicitis	32.1	4.2	2.2	2.0						
Australian	207.2	8.8	4.5	4.3						
Flag	315.2	26.7	5.1	6.1	4.0	2.1	2.7	3.6	1.1	1.6
Glass	295.8	14.0	1.4	3.9	1.0	2.8	1.9	2.8		
Ionosphere	63.9	7.9	3.0	4.9						
Iris	43.4	5.0	1.0	2.0	2.0					
Pima-diabetes	365.3	9.1	4.2	4.8						
Promotores	77.2	3.7	1.9	1.8						
Sonar	109.7	8.5	4.5	4.0						
Wine	63.3	4.0	1.0	2.0	1.0					

On simple data results are similar as from SVM (because they are almost optimal), but c3sep models are much simpler although only 3-sep. assumed.

Frameworks for Machine Learning

Similarity-based framework

Search for good models requires a frameworks to build and evaluate them.
$p(C_i|X;M)$ posterior classification probability or $y(X;M)$ approximators,
models M are parameterized in increasingly sophisticated way.
Similarity-Based Learning (SBL) or S-B Methods provide such framework.

(Dis)similarity:

- more general than feature-based description,
- no need for vector spaces (structured objects),
- more general than fuzzy approach (F-rules are reduced to P-rules),
- includes nearest neighbor algorithms, MLPs, RBFs, separable function networks, SVMs, kernel methods, specialized kernels, and many others!

A systematic search (greedy, beam), or evolutionary search in the space of all SBL models is used to select optimal combination of parameters & procedures, opening different types of optimization channels,
trying to discover appropriate bias for a given problem.

Result: several candidate models are created, already first very limited version gave best results in 7 out of 12 Stalog problems.

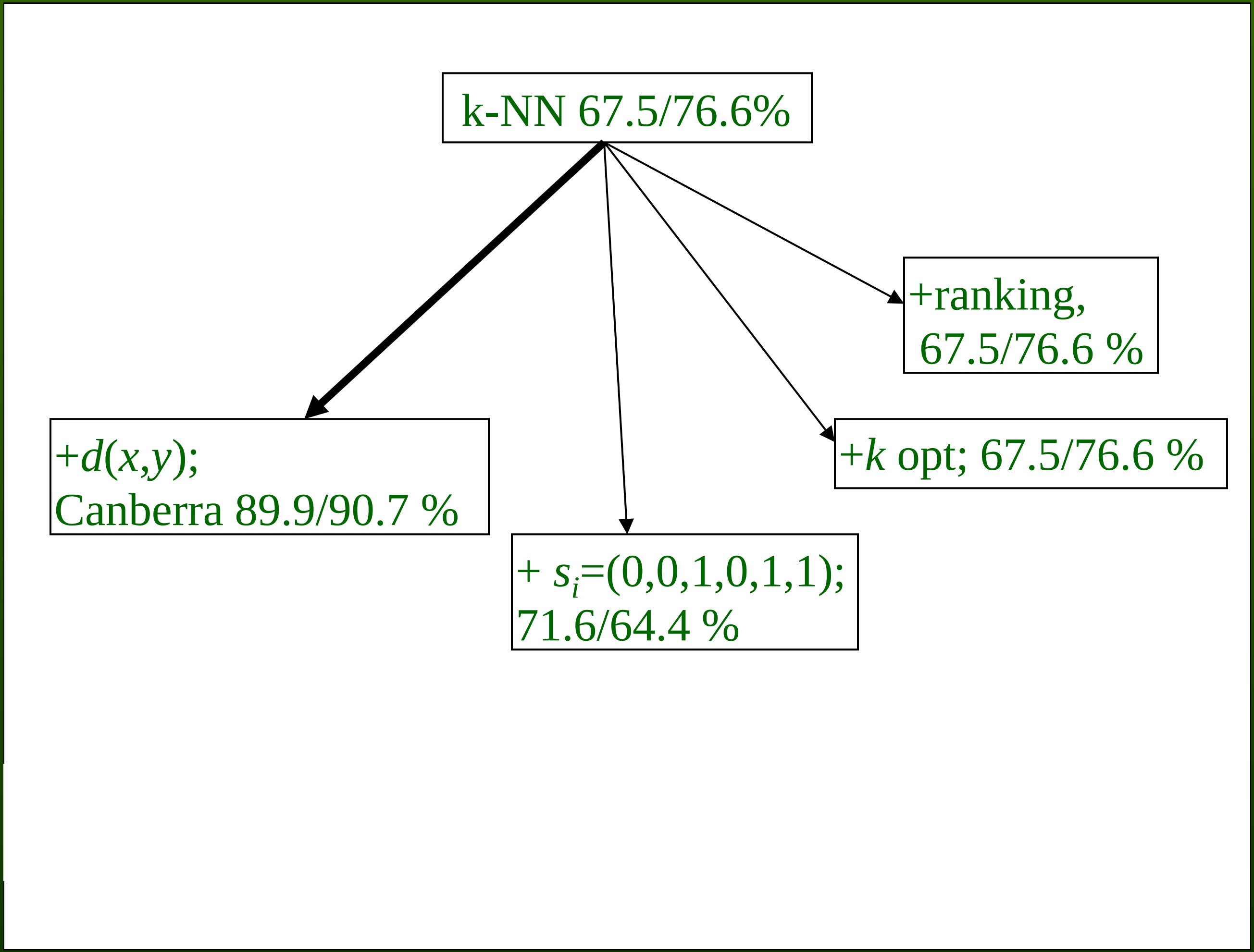
SBM framework components

- Pre-processing: objects O => features X, or (diss)similarities $D(O,O')$.
- Calculation of similarity between features $d(x_i,y_i)$ and objects $D(X,Y)$.
- Reference (or prototype) vector R selection/creation/optimization.
- Weighted influence of reference vectors $G(D(R_i,X))$, $i=1..k$.
- Functions/procedures to estimate $p(C|X;M)$ or approximator $y(X;M)$.
- Cost functions $E[D_T;M]$, various model selection/validation procedures.
- Optimization procedures for the whole model M_a.
- Search control procedures to create more complex models M_{a+1}.
- Creation of ensembles of (global, local, competent) models.

- $M=\{X(O), d(\cdot,\cdot), D(\cdot,\cdot), k, G(D), \{R\}, \{p_i(R)\}, E[\cdot], K(\cdot), S(\cdot,\cdot)\}$, where:
- $S(C_i,C_j)$ is a matrix evaluating similarity of the classes;
  a vector of observed probabilities $p_i(X)$ instead of hard labels.

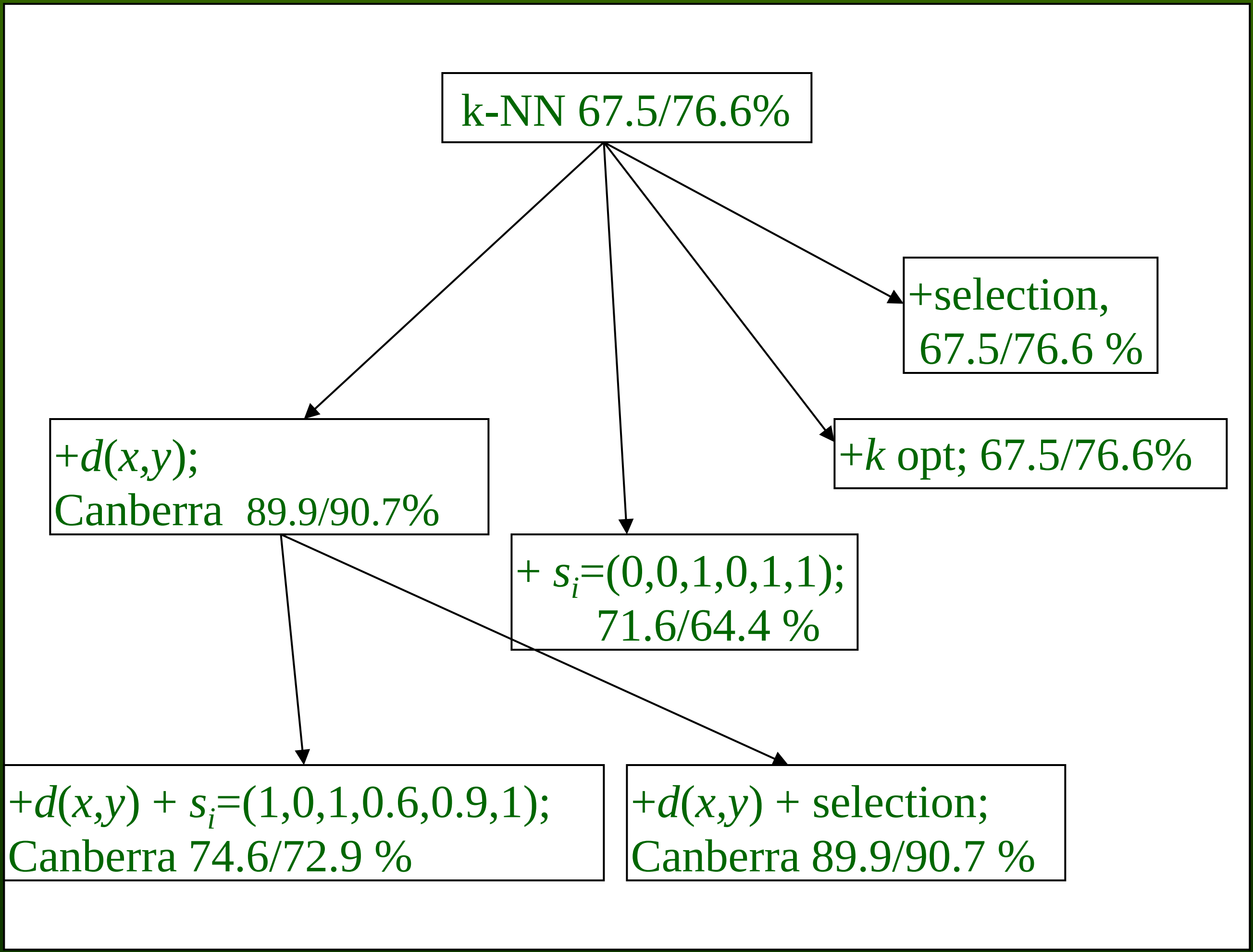
The kNN model $p(Ci|X;kNN) = p(C_i|X;k,D(\cdot),\{D_T\})$;
the RBF model: $p(Ci|X;RBF) = p(Ci|X;D(\cdot),G(D),\{R\})$,
MLP, SVM and many other models may all be “re-discovered” as a part of SBL.

Meta-learning in SBL scheme


k-NN 67.5/76.6%
+$d(x,y)$; Canberra 89.9/90.7 %
+ s_i=(0,0,1,0,1,1); 71.6/64.4 %
+k opt; 67.5/76.6 %
+ranking, 67.5/76.6 %

Start from kNN, k=1, all data & features, Euclidean distance, end with a new model based on novel combination of procedures and parameterizations.

Meta-learning in SBL scheme


k-NN 67.5/76.6%
+selection, 67.5/76.6 %
+$d(x,y)$; Canberra 89.9/90.7%
+k opt; 67.5/76.6%
+ s_i=(0,0,1,0,1,1); 71.6/64.4 %
+$d(x,y)$ + s_i=(1,0,1,0.6,0.9,1); Canberra 74.6/72.9 %
+$d(x,y)$ + selection; Canberra 89.9/90.7 %

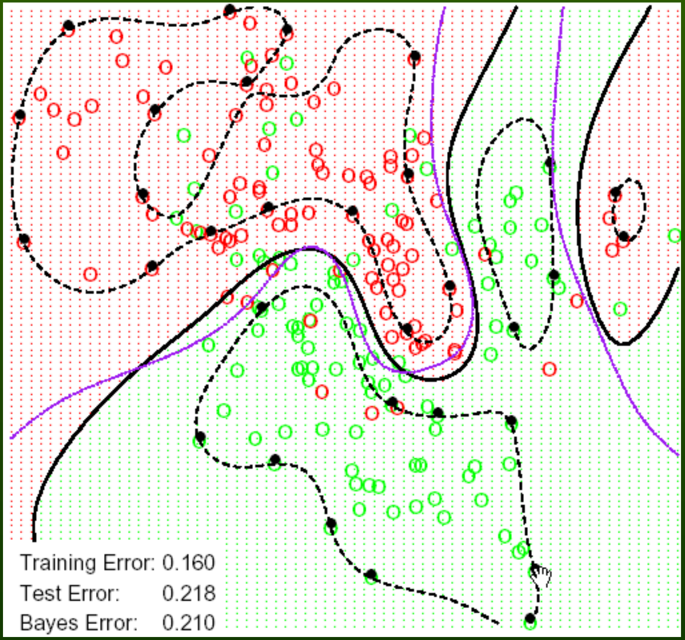
SBL program with many options developed by Karol Grudziński.

Kernels = similarity functions

Gaussian kernels in SVM: $z_i(X)=G(X;X_I,\sigma)$ radial features, X=>Z
Gaussian mixtures are close to optimal Bayesian errors. Solution requires continuous deformation of decision borders and is therefore rather easy.

Gaussian kernel, C=1.
In the kernel space Z decision borders are flat, but in the X space highly non-linear!

SVM is based on quadratic solver, without explicit features, but using Z features explicitly has some advantages:
Multiresolution (Locally Optimized Kernels): different σ for different support features, or using several kernels $z_i(X)=K(X;X_I,\sigma)$.
Use linear solvers, neural network, Naïve Bayes, or any other algorithm, all work fine.



Support Feature Machines (SFM): construct features based on projections, restricted linear combinations, kernel features, use feature selection.

Transformation-based framework

Find simplest model that is suitable for a given data, creating non-sep. that is easy to handle: simpler models generalize better, interpretation.

Compose transformations (neural layers), for example:

- Matching pursuit network for signal decomposition, QPC index.
- PCA network, with each node computing principal component.
- LDA nets, each node computes LDA direction (including FDA).
- ICA network, nodes computing independent components.
- KL, or Kullback-Leibler network with orthogonal or non-orthogonal components; max. of mutual information is a special case.
- c^2 and other statistical tests for dependency to aggregate features.
- Factor analysis network, computing common and unique factors.

Evolving Transformation Systems (Goldfarb 1990-2008), giving unified paradigm for inductive learning, structural processes as representations.

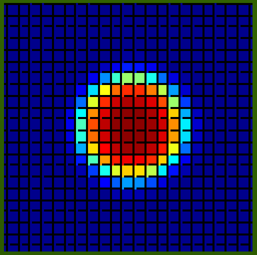
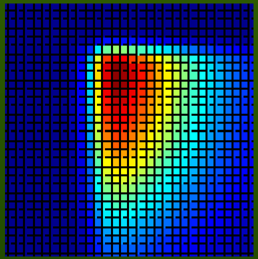
Heterogeneous systems

Next step: use components from different models.
Problems requiring different scales (multiresolution).

2-class problems, two situations:

C_1 inside the sphere, C_2 outside.
MLP: at least N+1 hyperplanes, $O(N^2)$ parameters.
RBF: 1 Gaussian, $O(N)$ parameters.
C_1 in the corner defined by (1,1 ... 1) hyperplane, C_2 outside.
MLP: 1 hyperplane, $O(N)$ parameters.
RBF: many Gaussians, $O(N^2)$ parameters, poor approx.
Combination: needs both hyperplane and hypersphere!

Logical rule: IF $x_1>0$ & $x_2>0$ THEN C_1 Else C_2
is not represented properly neither by MLP nor RBF!

Different types of functions in one model, first step beyond inspirations from single neurons => heterogeneous models are inspired by neural minicolumns, more complex information processing.

Heterogeneous everything

Homogenous systems: one type of “building blocks”, same type of decision borders, ex: neural networks, SVMs, decision trees, kNNs

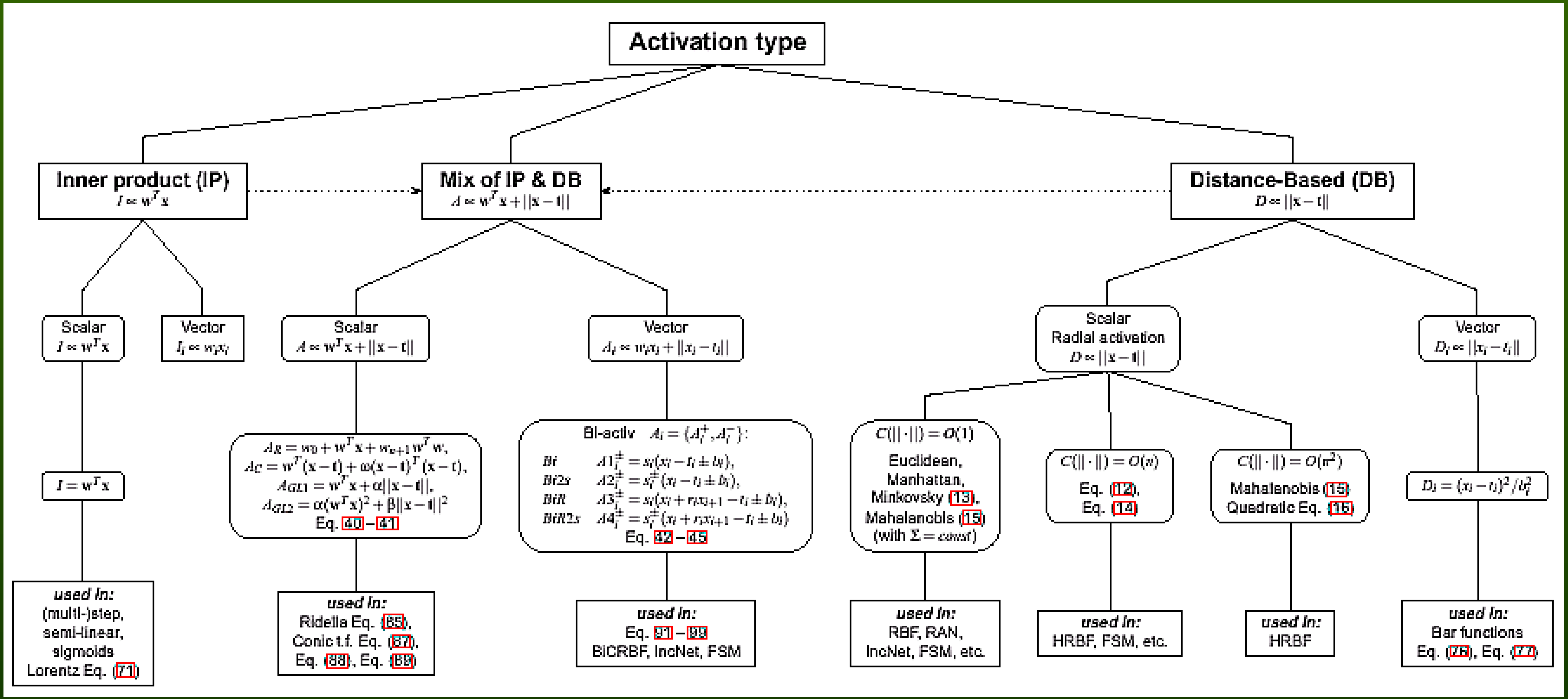
Committees combine many models together, but lead to complex models that are difficult to understand.

Ockham razor: simpler systems are better.
Discovering simplest class structures, inductive bias of the data, requires Heterogeneous Adaptive Systems (HAS).

HAS examples:
NN with different types of neuron transfer functions.
k-NN with different distance functions for each prototype.
Decision Trees with different types of test criteria.

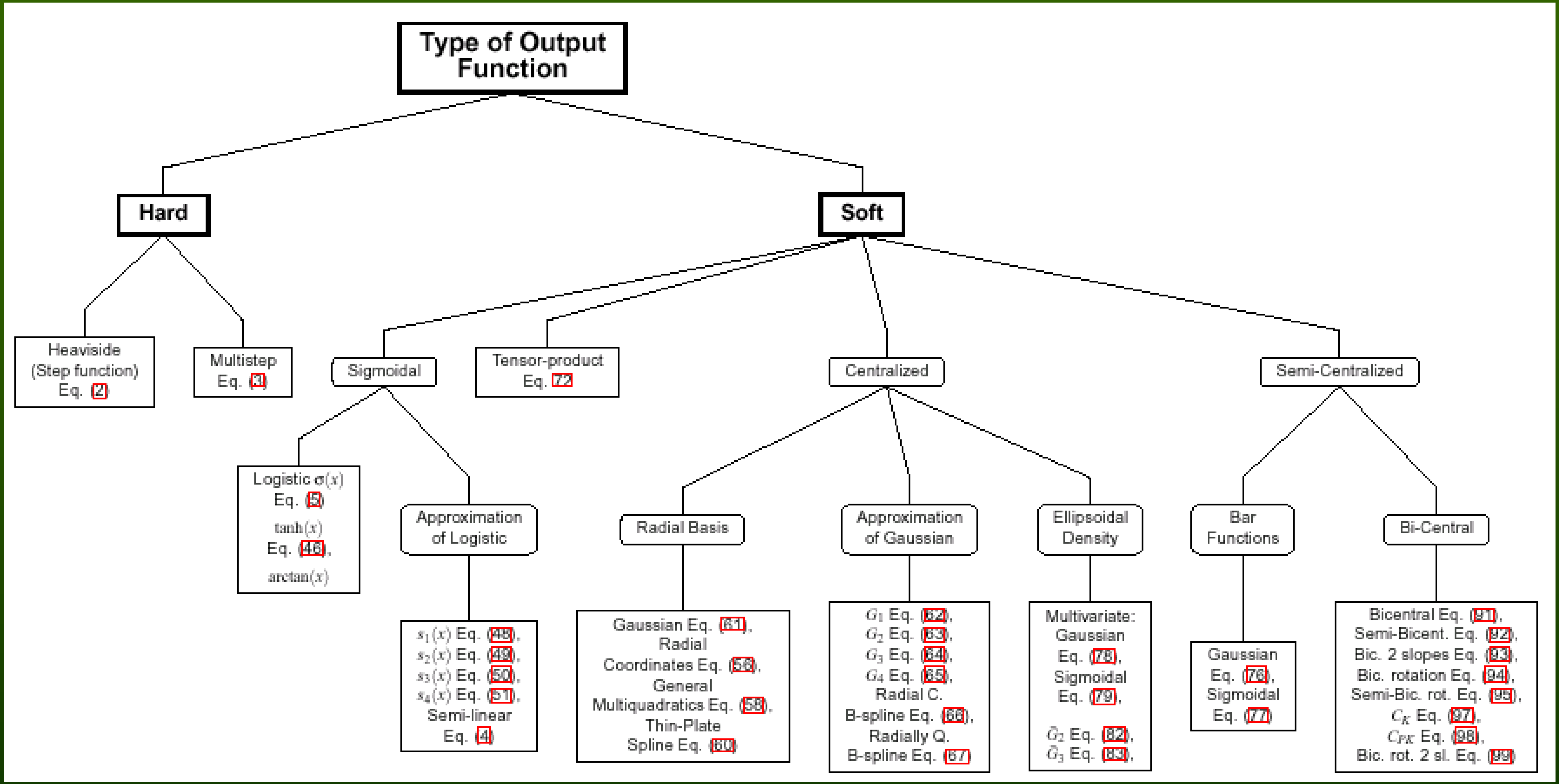
1. Start from large network, use regularization to prune.
2. Construct network adding nodes selected from a candidate pool.
3. Use very flexible functions, force them to specialize.

Taxonomy of NN activation functions


Activation type
Inner product (IP) $I \propto \mathbf{w}^T\mathbf{x}$
Mix of IP & DB $A \propto \mathbf{w}^T\mathbf{x} + ||\mathbf{x}-\mathbf{t}||$
Distance-Based (DB) $D \propto ||\mathbf{x}-\mathbf{t}||$
Scalar $I \propto \mathbf{w}^T\mathbf{x}$
Vector $I_i \propto w_i x_i$
Scalar $A \propto \mathbf{w}^T\mathbf{x} + ||\mathbf{x}-\mathbf{t}||$
Vector $A_i \propto w_i x_i + ||x_i - t_i||$
Scalar Radial activation $D \propto ||\mathbf{x}-\mathbf{t}||$
Vector $D_i \propto ||x_i - t_i||$
$I = \mathbf{w}^T\mathbf{x}$
$A_R = w_0 + \mathbf{w}^T\mathbf{x} + w_{n+1}\mathbf{w}^T\mathbf{w}$, $A_C = \mathbf{w}^T(\mathbf{x}-\mathbf{t}) + \omega(\mathbf{x}-\mathbf{t})^T(\mathbf{x}-\mathbf{t})$, $A_{GL1} = \mathbf{w}^T\mathbf{x} + \alpha||\mathbf{x}-\mathbf{t}||$, $A_{GL2} = \alpha(\mathbf{w}^T\mathbf{x})^2 + \beta||\mathbf{x}-\mathbf{t}||^2$ Eq. 40 – 41
Bi-activ $A_i = \{A_i^+, A_i^-\}$:
Bi $A1_i^\pm = s_i(x_i - t_i \pm b_i)$,
Bi2s $A2_i^\pm = s_i^\pm(x_i - t_i \pm b_i)$,
BiR $A3_i^\pm = s_i(x_i + r_i x_{i+1} - t_i \pm b_i)$,
BiR2s $A4_i^\pm = s_i^\pm(x_i + r_i x_{i+1} - t_i \pm b_i)$
Eq. 42 – 45
$C(||\cdot||) = O(1)$ Euclidean, Manhattan, Minkovsky (13), Mahalanobis (15) (with $\Sigma = const$)
$C(||\cdot||) = O(n)$ Eq. (12), Eq. (14)
$C(||\cdot||) = O(n^2)$ Mahalanobis (15) Quadratic Eq. (16)
$D_i = (x_i - t_i)^2/b_i^2$
used in: (multi-)step, semi-linear, sigmoids Lorentz Eq. (71)
used in: Ridella Eq. (65), Conic t.f. Eq. (67), Eq. (68), Eq. (69)
used in: Eq. 91 – 93 BiCRBF, IncNet, FSM
used in: RBF, RAN, IncNet, FSM, etc.
used in: HRBF, FSM, etc.
used in: HRBF
used in: Bar functions Eq. (76), Eq. (77)

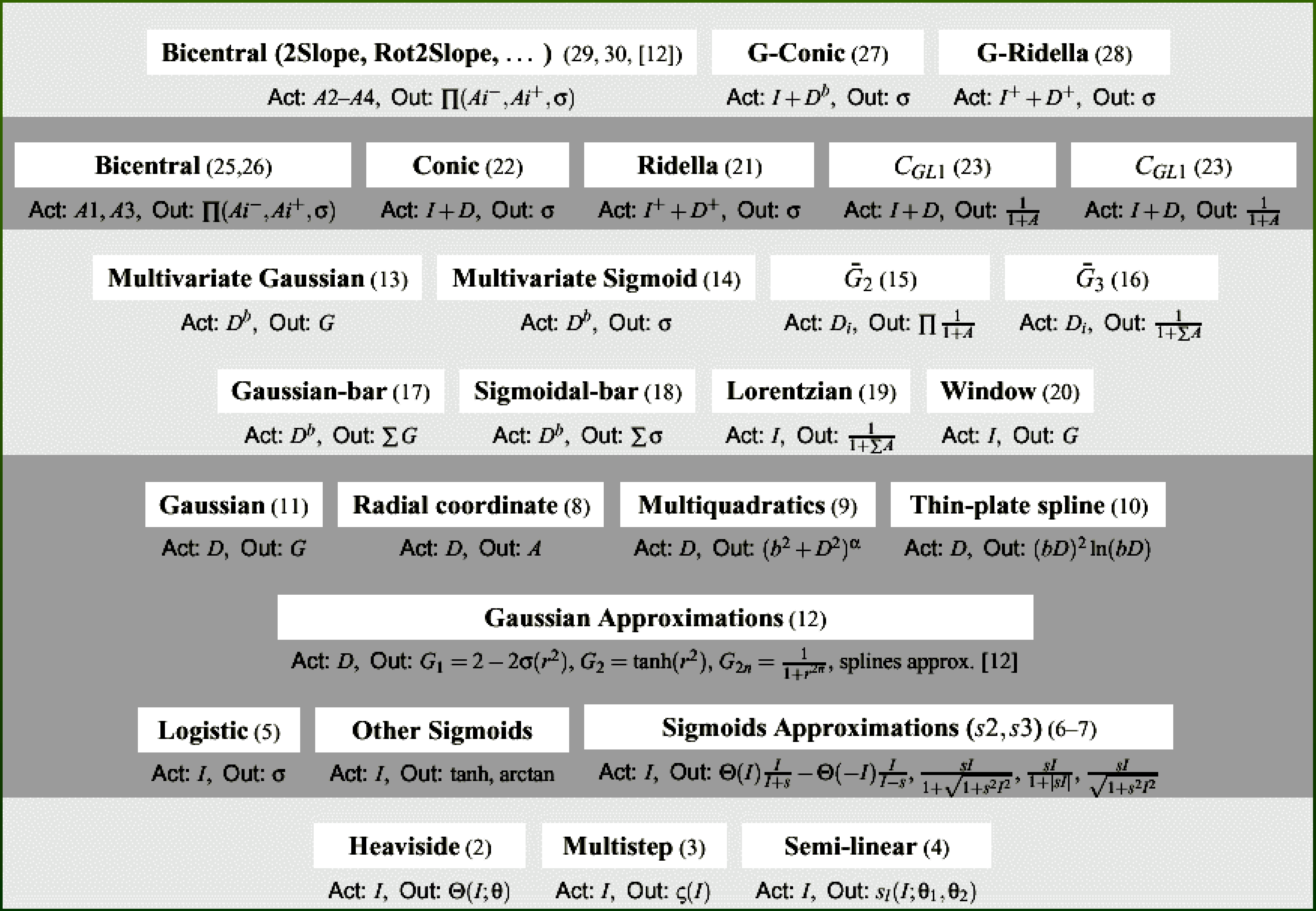
Duch W, Jankowski N (1999) Survey of neural transfer functions, Neural Computing Surveys 2: 163-213, now ~300 citations and growing.

Taxonomy of NN output functions


Type of Output Function
Hard
Soft
Heaviside (Step function) Eq. (2)
Multistep Eq. (3)
Sigmoidal
Tensor-product Eq. 72
Centralized
Semi-Centralized
Logistic $\sigma(x)$ Eq. (5) $\tanh(x)$ Eq. (46), $\arctan(x)$
Approximation of Logistic
$s_1(x)$ Eq. (48), $s_2(x)$ Eq. (49), $s_3(x)$ Eq. (50), $s_4(x)$ Eq. (51), Semi-linear Eq. (4)
Radial Basis
Gaussian Eq. (61), Radial Coordinates Eq. (56), General Multiquadratics Eq. (58), Thin-Plate Spline Eq. (60)
Approximation of Gaussian
G_1 Eq. (62), G_2 Eq. (63), G_3 Eq. (64), G_4 Eq. (65), Radial C. B-spline Eq. (66), Radially Q. B-spline Eq. (67)
Ellipsoidal Density
Multivariate: Gaussian Eq. (78), Sigmoidal Eq. (79), $\bar{G}_2$ Eq. (82), $\bar{G}_3$ Eq. (83),
Bar Functions
Gaussian Eq. (76), Sigmoidal Eq. (77)
Bi-Central
Bicentral Eq. (91), Semi-Bicent. Eq. (92), Bic. 2 slopes Eq. (93), Bic. rotation Eq. (94), Semi-Bic. rot. Eq. (95), C_K Eq. (97), C_{PK} Eq. (98), Bic. rot. 2 sl. Eq. (99)

Perceptron: implements logical rule $x>\theta$ for x with Gaussian uncertainty.

Taxonomy - TF



Bicentral (2Slope, Rot2Slope, …) (29, 30, [12])
Act: $A2$–$A4$, Out: $\prod(Ai^-, Ai^+, \sigma)$

G-Conic (27)
Act: $I + D^b$, Out: σ

G-Ridella (28)
Act: $I^+ + D^+$, Out: σ

Bicentral (25,26)
Act: $A1, A3$, Out: $\prod(Ai^-, Ai^+, \sigma)$

Conic (22)
Act: $I + D$, Out: σ

Ridella (21)
Act: $I^+ + D^+$, Out: σ

C_{GL1} (23)
Act: $I + D$, Out: $\frac{1}{1+A}$

C_{GL1} (23)
Act: $I + D$, Out: $\frac{1}{1+A}$

Multivariate Gaussian (13)
Act: D^b, Out: G

Multivariate Sigmoid (14)
Act: D^b, Out: σ

$\bar{G}_2$ (15)
Act: D_i, Out: $\prod \frac{1}{1+A}$

$\bar{G}_3$ (16)
Act: D_i, Out: $\frac{1}{1+\sum A}$

Gaussian-bar (17)
Act: D^b, Out: $\sum G$

Sigmoidal-bar (18)
Act: D^b, Out: $\sum \sigma$

Lorentzian (19)
Act: I, Out: $\frac{1}{1+\sum A}$

Window (20)
Act: I, Out: G

Gaussian (11)
Act: D, Out: G

Radial coordinate (8)
Act: D, Out: A

Multiquadratics (9)
Act: D, Out: $(b^2 + D^2)^\alpha$

Thin-plate spline (10)
Act: D, Out: $(bD)^2 \ln(bD)$

Gaussian Approximations (12)
Act: D, Out: $G_1 = 2 - 2\sigma(r^2)$, $G_2 = \tanh(r^2)$, $G_{2n} = \frac{1}{1+r^{2n}}$, splines approx. [12]

Logistic (5)
Act: I, Out: σ

Other Sigmoids
Act: I, Out: tanh, arctan

Sigmoids Approximations $(s2, s3)$ (6–7)
Act: I, Out: $\Theta(I)\frac{I}{I+s} - \Theta(-I)\frac{I}{I-s}$, $\frac{sI}{1+\sqrt{1+s^2I^2}}$, $\frac{sI}{1+|sI|}$, $\frac{sI}{\sqrt{1+s^2I^2}}$

Heaviside (2)
Act: I, Out: $\Theta(I;\theta)$

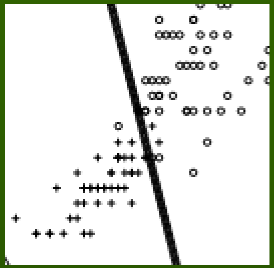
Multistep (3)
Act: I, Out: $\varsigma(I)$

Semi-linear (4)
Act: I, Out: $s_l(I;\theta_1,\theta_2)$

HAS decision trees

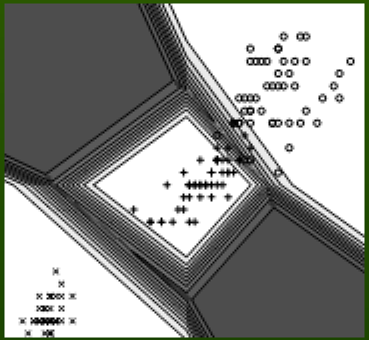
Decision trees select the best feature/threshold value for univariate and multivariate trees:

$$X_i < \theta_k \text{ or } T(\mathbf{X};\mathbf{W},\theta_k) = \sum_i W_i X_i < \theta_k$$



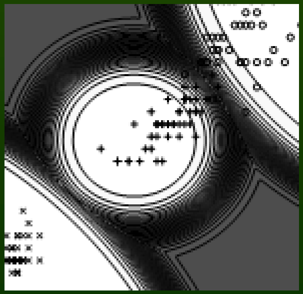
Decision borders: hyperplanes.

Introducing tests based on L_α Minkovsky metric.



$$T(\mathbf{X};\mathbf{R},\theta_R) = \|\mathbf{X}-\mathbf{R}\|_\alpha = \sum_i |X_i - R_i|^{1/\alpha} < \theta_R$$

Such DT use radial kernel features!



For L_2 spherical decision border are produced.

For L_∞ rectangular border are produced.

For large databases first clusterize data to get candidate references R.

SSV HAS DT example

SSV HAS tree in GhostMiner 3.0, Wisconsin breast cancer (UCI)
699 cases, 9 features (cell parameters, 1..10)
Classes: benign 458 (65.5%) & malignant 241 (34.5%).

Single rule gives simplest known description of this data:

IF $||X-R_{303}|| < 20.27$ then malignant

else benign coming most often in 10xCV

Accuracy = 97.4%, good prototype for malignant case!

Gives simple thresholds, that's what MDs like the most!

Best 10CV around 97.5±1.8% (Naïve Bayes + kernel, or opt. SVM)

SSV without distances: 96.4±2.1%

C 4.5 gives 94.7±2.0%

Several simple rules of similar accuracy but different specificity or sensitivity may be created using HAS DT.
Need to select or weight features and select good prototypes.

Support Feature Machines

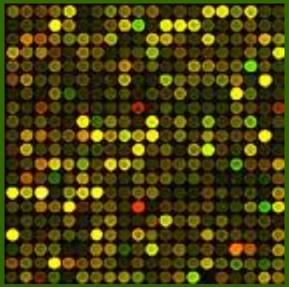
General principle: complementarity of information processed by parallel interacting streams with hierarchical organization (Grossberg, 2000).

Cortical minicolumns provide various features for higher processes.

Create information that is easily used by various ML algorithms: explicitly build enhanced space adding more transformations.

- X , original features
- Z=WX, random linear projections, other projections (PCA< ICA, PP)
- Q = optimized Z using Quality of Projected Clusters or other PP techniques.
- $H=[Z_1,Z_2]$, intervals containing pure clusters on projections.
- $K=K(X,X_i)$, kernel features.
- $HK=[K_1,K_2]$, intervals on kernel features

Kernel-based SVM is equivalent to linear SVM in the explicitly constructed kernel space, enhancing this space leads to improvement of results.
LDA is one option, but many other algorithms benefit from information in enhanced feature spaces; best results in various combination X+Z+Q+H+K+HK.



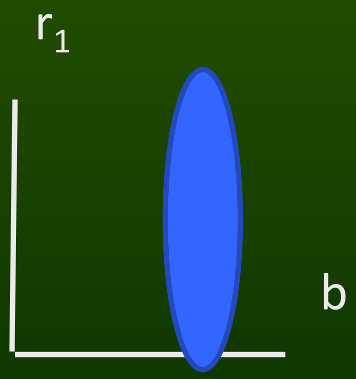
Binary features

Binary features:

- B1: unrestricted projections;
  MAP classifiers, $p(C|b)$; $2N_C$ regions, complexity $O(1)$
- B2: Binary: restricted by other binary features;
  complexes $b_1 \wedge b_2 \ldots \wedge b_k$; complexity $O(2^k)$
- B3: Binary: restricted by distance;
  $b \wedge r_1 \in [r_{1-}, r_{1+}] \ldots \wedge r_k \in [r_{k-}, r_{k+}]$;
  separately for each b value.

- Ex: $b=1$, $r_1 \in [0, 1]$
  take vectors only from this slice

N1: Nominal – like binary.



Datasets

Dataset	#Features	#Samples	#Samples per class		
Australian	15	690	383 no	307 yes	
Appendicitis	7	106	85 C_1	21 C_2	
Heart	13	303	164 absence	139 presence	
Diabetes	8	768	268 C_1	500 C_2	
Wisconsin	9	699	458 benign	241 malignant	
Hypothyroid	21	3772	93 C_1	191 C_2	3488 C_3

B1/B2 Features

Dataset	B1/B2 Features	
Australian	$F8 < 0.5$	$F8 \geq 0.5 \wedge F9 \geq 0.5$
Appendicitis	$F7 \geq 7520.5$	$F7 < 7520.5 \wedge F4 < 12$
Heart	$F13 < 4.5 \wedge F12 < 0.5$	$F13 \geq 4.5 \wedge F3 \geq 3.5$
Diabetes	$F2 < 123.5$	$F2 \geq 143.5$
Wisconsin	$F2 < 2.5$	$F2 \geq 4.5$
Hypothyroid	$F17 < 0.00605$	$F17 \geq 0.00605 \wedge F21 < 0.06472$

Example of B1 features taken from important segments of decision trees.
These features used in various learning systems greatly simplify their models and increase their accuracy.
Convert Decision Tree to Distance Functions for more!
Extending training vectors with these features makes almost all learning algorithms reach similar high accuracy!
Other features that frequently proved useful on such data: P1 prototypes.

Description of new features

X - original features

K - kernel features (Gaussian local kernels)

Z - unrestricted linear projections

H - restricted (clustered) projections

15 feature spaces based on combinations of these 4 different type of features may be constructed to extend original space: X, K, Z, H, K+Z, K+H, Z+H, K+Z+H, X+K, X+Z, X+H, X+K+Z, X+K+H, X+Z+H, X+K+Z+H.

The final vector *X* is thus composed from a number of
$X = [x_1..x_n, z_1.., h_1.., k_1..]$ features.
In the SF space linear discrimination is used (SVML),
although other methods may find better solution.

Only a few results are presented here (big table).

SFM vs SVM

SFM generalize SVM approach by explicitly building feature space: enhance your input space adding kernel features $z_i(X)=K(X;SV_i)$

+ any other useful types of features.

SFM advantages comparing to SVM:

- Kernel-based SVM ⇔ SVML in explicitly constructed kernel space, allows for various feature selection methods, local kernel optimization.
- Extend input + kernel space => improvement.
- Linear discrimination on explicit representation of features = easy interpretation of SFM functions as combination of local similarity evaluations (biologically plausible).

SFM vs SVM

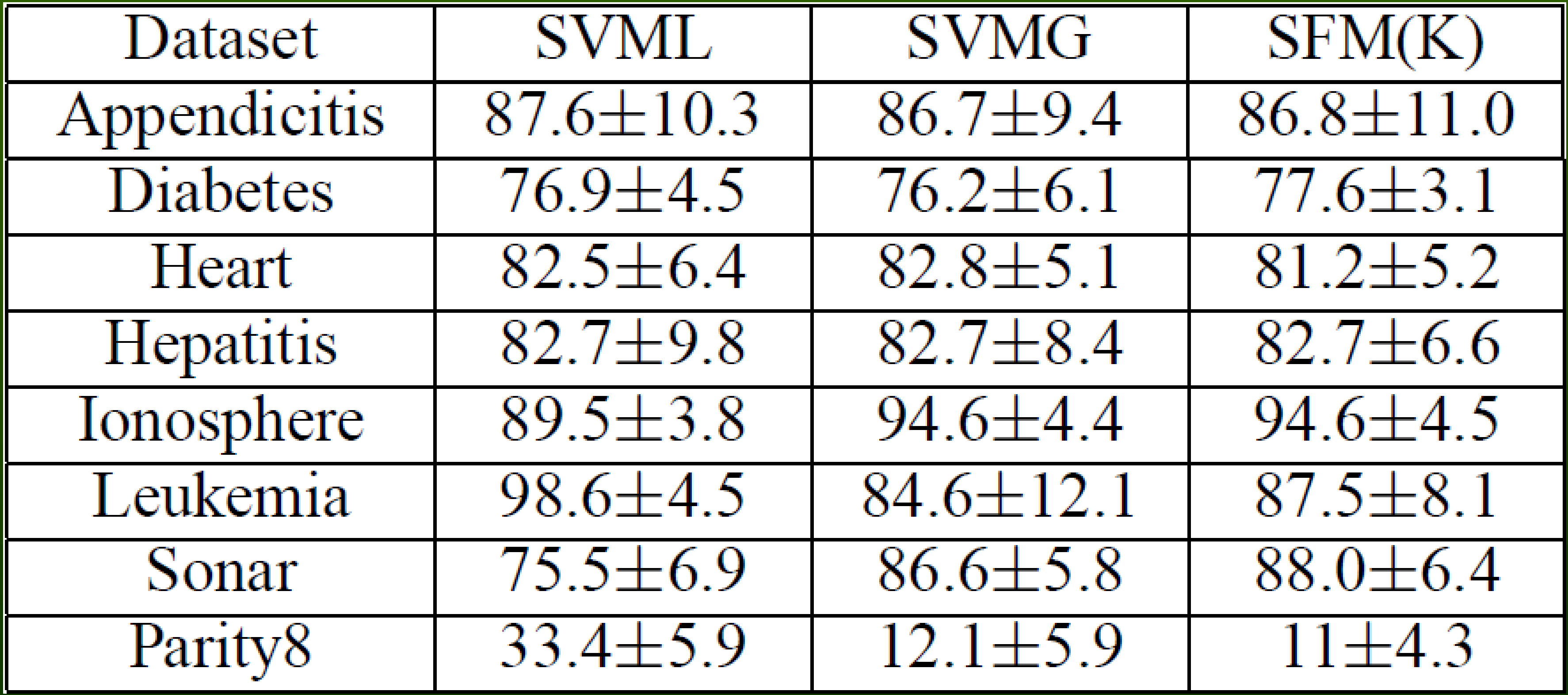
How to extend the feature space, creating SF space?

- Use various kernels with various parameters.
- Use global features obtained from various projections.
- Use local features to handle exceptions.
- Use feature selection to define optimal support feature space.

Many algorithms may be used in SF space to generate the final solution.

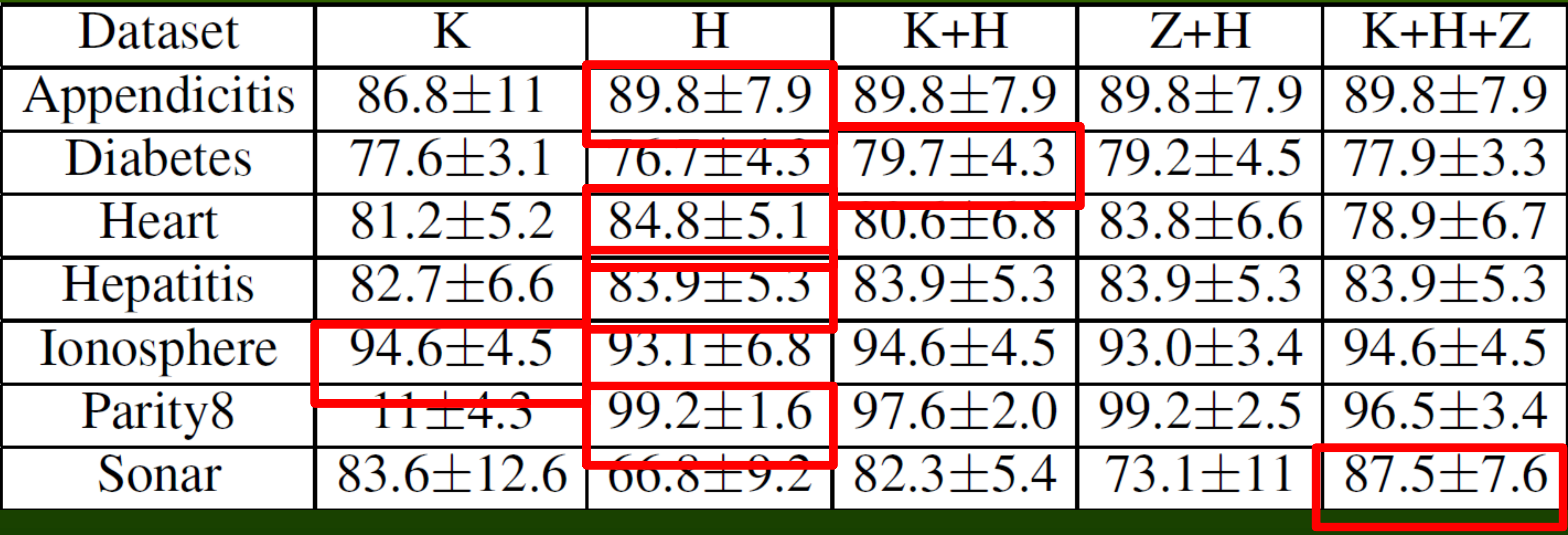
In the current version three types of features are used, but many extensions are possible.

Results
SVM vs SFM in the kernel space only



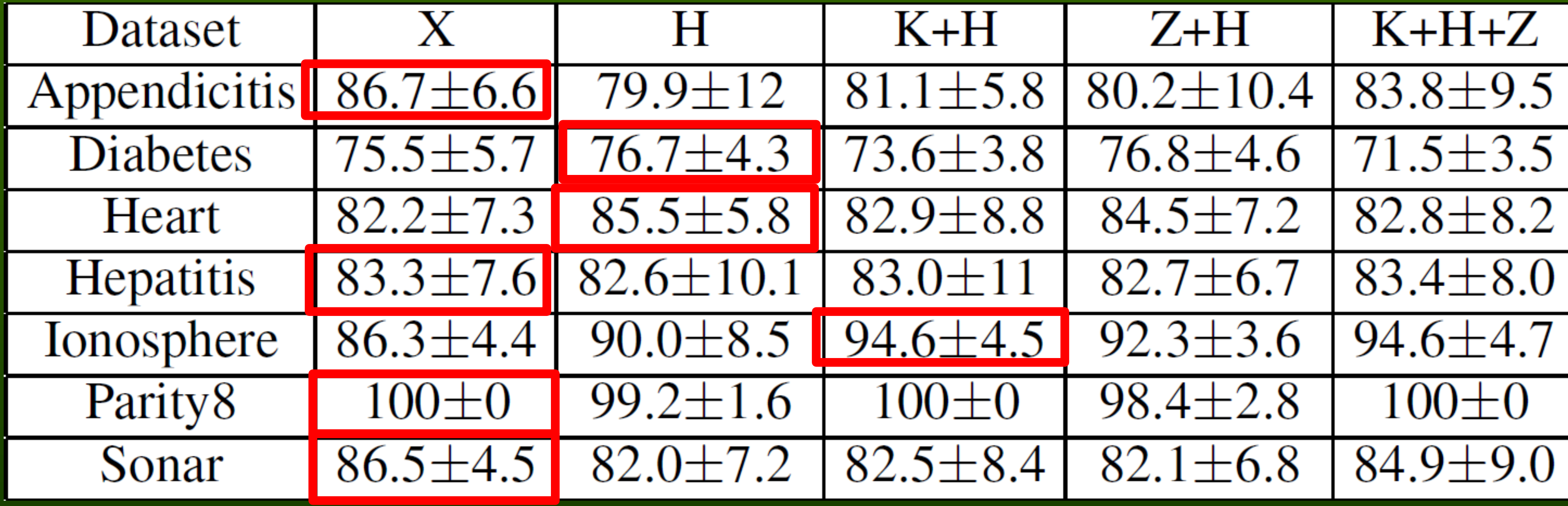
Dataset	SVML	SVMG	SFM(K)
Appendicitis	87.6±10.3	86.7±9.4	86.8±11.0
Diabetes	76.9±4.5	76.2±6.1	77.6±3.1
Heart	82.5±6.4	82.8±5.1	81.2±5.2
Hepatitis	82.7±9.8	82.7±8.4	82.7±6.6
Ionosphere	89.5±3.8	94.6±4.4	94.6±4.5
Leukemia	98.6±4.5	84.6±12.1	87.5±8.1
Sonar	75.5±6.9	86.6±5.8	88.0±6.4
Parity8	33.4±5.9	12.1±5.9	11±4.3

Results: SFM in extended spaces



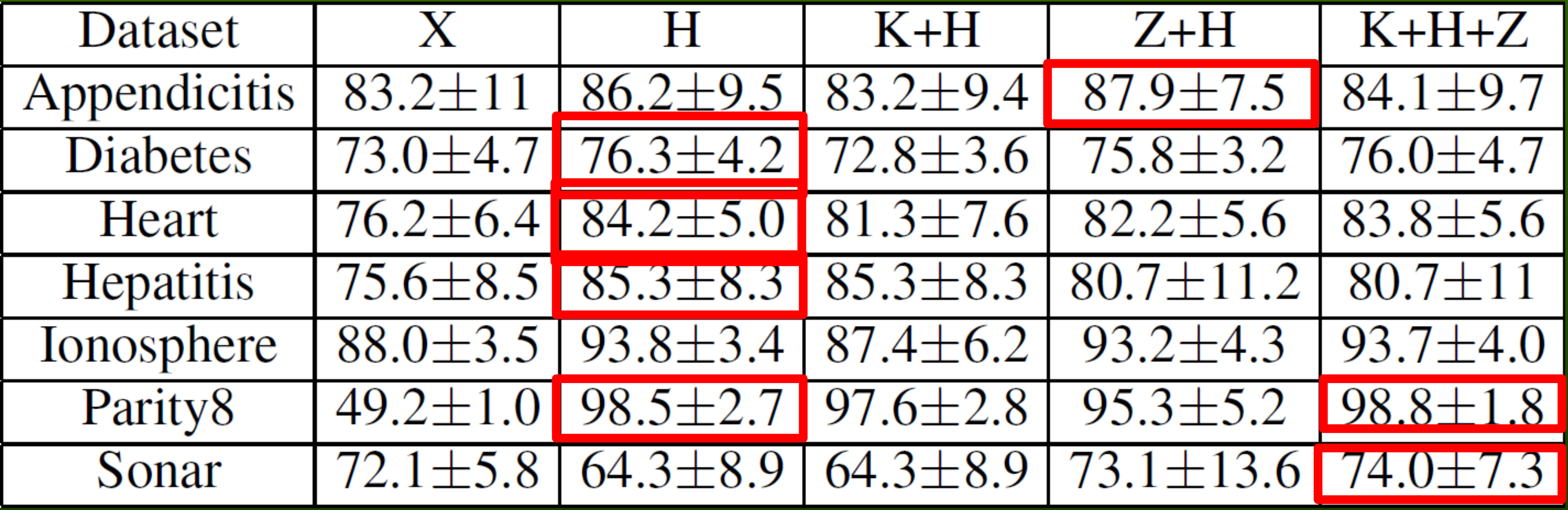
Dataset	K	H	K+H	Z+H	K+H+Z
Appendicitis	86.8±11	89.8±7.9	89.8±7.9	89.8±7.9	89.8±7.9
Diabetes	77.6±3.1	76.7±4.3	79.7±4.3	79.2±4.5	77.9±3.3
Heart	81.2±5.2	84.8±5.1	80.6±6.8	83.8±6.6	78.9±6.7
Hepatitis	82.7±6.6	83.9±5.3	83.9±5.3	83.9±5.3	83.9±5.3
Ionosphere	94.6±4.5	93.1±6.8	94.6±4.5	93.0±3.4	94.6±4.5
Parity8	11±4.3	99.2±1.6	97.6±2.0	99.2±2.5	96.5±3.4
Sonar	83.6±12.6	66.8±9.2	82.3±5.4	73.1±11	87.5±7.6

Results: kNN in extended spaces



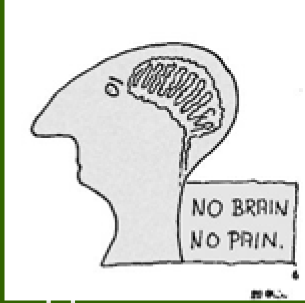
Dataset	X	H	K+H	Z+H	K+H+Z
Appendicitis	86.7±6.6	79.9±12	81.1±5.8	80.2±10.4	83.8±9.5
Diabetes	75.5±5.7	76.7±4.3	73.6±3.8	76.8±4.6	71.5±3.5
Heart	82.2±7.3	85.5±5.8	82.9±8.8	84.5±7.2	82.8±8.2
Hepatitis	83.3±7.6	82.6±10.1	83.0±11	82.7±6.7	83.4±8.0
Ionosphere	86.3±4.4	90.0±8.5	94.6±4.5	92.3±3.6	94.6±4.7
Parity8	100±0	99.2±1.6	100±0	98.4±2.8	100±0
Sonar	86.5±4.5	82.0±7.2	82.5±8.4	82.1±6.8	84.9±9.0

Results: SSV in extended spaces



Dataset	X	H	K+H	Z+H	K+H+Z
Appendicitis	83.2±11	86.2±9.5	83.2±9.4	87.9±7.5	84.1±9.7
Diabetes	73.0±4.7	76.3±4.2	72.8±3.6	75.8±3.2	76.0±4.7
Heart	76.2±6.4	84.2±5.0	81.3±7.6	82.2±5.6	83.8±5.6
Hepatitis	75.6±8.5	85.3±8.3	85.3±8.3	80.7±11.2	80.7±11
Ionosphere	88.0±3.5	93.8±3.4	87.4±6.2	93.2±4.3	93.7±4.0
Parity8	49.2±1.0	98.5±2.7	97.6±2.8	95.3±5.2	98.8±1.8
Sonar	72.1±5.8	64.3±8.9	64.3±8.9	73.1±13.6	74.0±7.3

Learning from others …



Learn to transfer knowledge by extracting interesting features created by different systems. Ex. prototypes, combinations of features with thresholds …

=> Universal Learning Machines.

Classify all types of features – what type of info they extract?

B1: Binary – unrestricted projections b_1
B2: Binary – complexes $b_1 \wedge b_2 \ldots \wedge b_k$

B3: Binary – restricted by distance $(b_i = 0) \wedge r_1 \in [r_1^-, r_1^+] \wedge r_2 \in [r_2^-, r_2^+] \ldots$

R1: Line – original real features r_i; non-linear thresholds for "contrast enhancement" $\sigma(r_i - b_i)$; intervals (k-sep).

R4: Line – restricted by distance, original feature; thresholds; intervals (k-sep); more general 1D patterns.

P1: Prototypes: general q-separability, weighted distance functions or specialized kernels.

M1: Motifs, based on correlations between elements rather than input values.

SFM conclusions

- SFM is focused on generation of new features, rather than optimization and improvement of classifiers.
- SFM may be seen as mixture of experts; each expert is a simple model based on single feature: projection, localized projection, optimized projection, various kernel features.
- For different data different types of features may be important => no universal set of features, but easy to test and select.

SFM conclusions

- Kernel-based SVM is equivalent to the use of kernel features combined with LD.
- Mixing different kernels and different types of features: better feature space than single-kernel solution.
- Complex data require decision borders with different complexity. SFM offers multiresolution (ex: different dispersions for every SV).
- Kernel-based learning implicitly project data into high-dimensional space, creating there flat decision borders and facilitating separability.

Universal Learning Machines

ULM is composed from two main modules:

- feature constructors,
- simple classifiers.

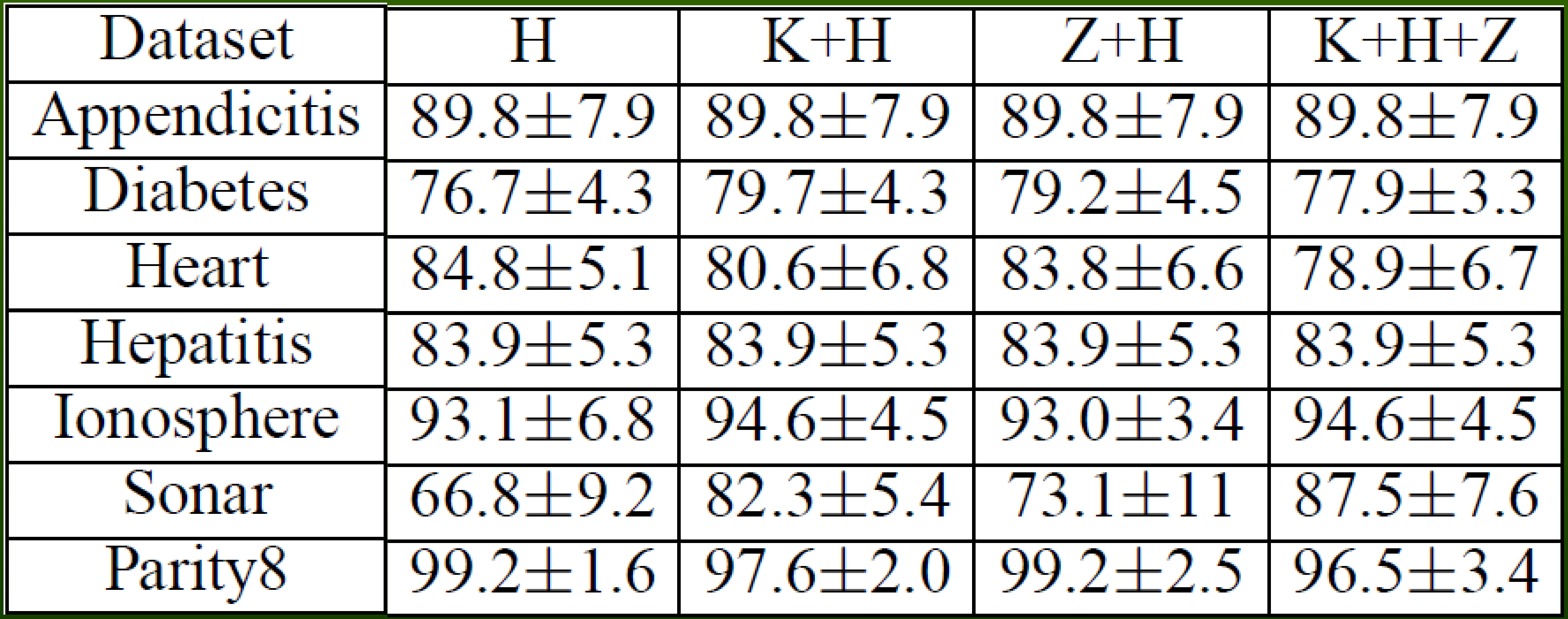
In machine learning features are used to calculate:

- linear combinations of feature values,
- calculate distances (dissimilarites), scaled (includes selection)

Is this sufficient?

- No, non-linear functions of features carry information that cannot be easily recovered by CI methods.
- Kernel approaches: linear solutions in the kernel space, implicitly add new features based on similarity $K(X,S_V)$.
- ULM idea: create potentially useful, redundant set of futures.
  How? Learn what other models do well! Implement transfer learning.

Results
(SFM in extended spaces)

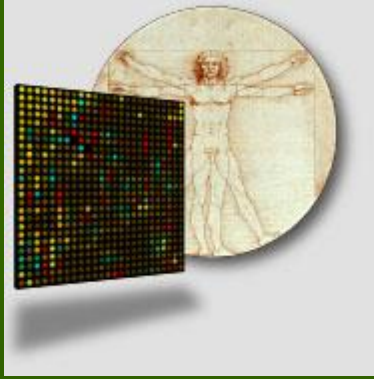


Dataset	H	K+H	Z+H	K+H+Z
Appendicitis	89.8±7.9	89.8±7.9	89.8±7.9	89.8±7.9
Diabetes	76.7±4.3	79.7±4.3	79.2±4.5	77.9±3.3
Heart	84.8±5.1	80.6±6.8	83.8±6.6	78.9±6.7
Hepatitis	83.9±5.3	83.9±5.3	83.9±5.3	83.9±5.3
Ionosphere	93.1±6.8	94.6±4.5	93.0±3.4	94.6±4.5
Sonar	66.8±9.2	82.3±5.4	73.1±11	87.5±7.6
Parity8	99.2±1.6	97.6±2.0	99.2±2.5	96.5±3.4

$K=K(X,X_i)$ $Z=WX$ $H=[Z_1,Z_2]$

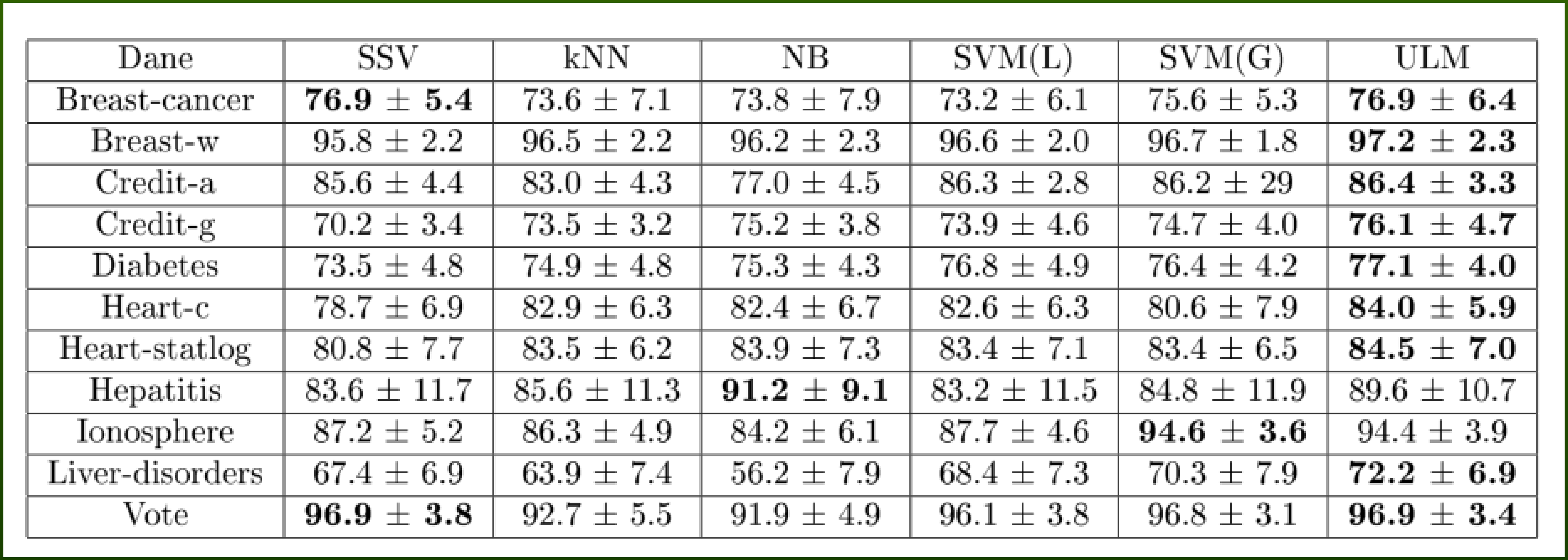
Dataset	Classifier		
	SVM (#SV)	SSV (#Leafs)	NB
Australian	84.9±5.6 (203)	84.9±3.9 (4)	80.3±3.8
ULM	86.8±5.3(166)	87.1±2.5(4)	85.5±3.4
Features	B1(2) + P1(3)	B1(2) + R1(1) + P1(3)	B1(2)
Appendicitis	87.8±8.7 (31)	88.0±7.4 (4)	86.7±6.6
ULM	91.4±8.2(18)	91.7±6.7(3)	91.4±8.2
Features	B1(2)	B1(2)	B1(2)
Heart	82.1±6.7 (101)	76.8±9.6 (6)	84.2±6.1
ULM	83.4±3.5(98)	79.2±6.3(6)	84.5±6.8
Features	Data + R1(3)	Data + R1(3)	Data + B1(2)
Diabetes	77.0±4.9 (361)	73.6±3.4 (4)	75.3±4.7
ULM	78.5±3.6(338)	75.0±3.3(3)	76.5±2.9
Features	Data + R1(3) + P1(4)	B1(2)	Data + B1(2)
Wisconsin	96.6±1.6 (46)	95.2±1.5 (8)	96.0±1.5
ULM	97.2±1.8(45)	97.4±1.6(2)	97.2±2.0
Features	Data + R1(1) + P1(4)	R1(1)	R1(1)
Hypothyroid	94.1±0.6 (918)	99.7±0.5 (12)	41.3±8.3
ULM	99.5±0.4(80)	99.6±0.4(8)	98.1±0.7
Features	Data + B1(2)	Data + B1(2)	Data + B1(2)

ULM conclusions



- Systematic explorations of features and transformations allows for discovery of simple models that more sophisticated learning systems may miss; results always improve and models simplify!
- Some benchmark problems have been found rather trivial, and have been solved with a single binary feature, one constrained nominal feature, or one new feature constructed as a projection on a line connecting means of two classes – always try simplest methods!
- Kernel-based features offer an attractive alternative to current kernel-based SVM approaches, offering multiresolution and adaptive regularization possibilities, combined with LDA or SVNT.
- Analysis of images, multimedia streams or biosequences may require even more sophisticated ways of constructing features starting from available input features.
- Learn from others, not only on your own errors!

Universal Learning Machines



Dane	SSV	kNN	NB	SVM(L)	SVM(G)	ULM
Breast-cancer	**76.9 ± 5.4**	73.6 ± 7.1	73.8 ± 7.9	73.2 ± 6.1	75.6 ± 5.3	**76.9 ± 6.4**
Breast-w	95.8 ± 2.2	96.5 ± 2.2	96.2 ± 2.3	96.6 ± 2.0	96.7 ± 1.8	**97.2 ± 2.3**
Credit-a	85.6 ± 4.4	83.0 ± 4.3	77.0 ± 4.5	86.3 ± 2.8	86.2 ± 29	**86.4 ± 3.3**
Credit-g	70.2 ± 3.4	73.5 ± 3.2	75.2 ± 3.8	73.9 ± 4.6	74.7 ± 4.0	**76.1 ± 4.7**
Diabetes	73.5 ± 4.8	74.9 ± 4.8	75.3 ± 4.3	76.8 ± 4.9	76.4 ± 4.2	**77.1 ± 4.0**
Heart-c	78.7 ± 6.9	82.9 ± 6.3	82.4 ± 6.7	82.6 ± 6.3	80.6 ± 7.9	**84.0 ± 5.9**
Heart-statlog	80.8 ± 7.7	83.5 ± 6.2	83.9 ± 7.3	83.4 ± 7.1	83.4 ± 6.5	**84.5 ± 7.0**
Hepatitis	83.6 ± 11.7	85.6 ± 11.3	**91.2 ± 9.1**	83.2 ± 11.5	84.8 ± 11.9	89.6 ± 10.7
Ionosphere	87.2 ± 5.2	86.3 ± 4.9	84.2 ± 6.1	87.7 ± 4.6	**94.6 ± 3.6**	94.4 ± 3.9
Liver-disorders	67.4 ± 6.9	63.9 ± 7.4	56.2 ± 7.9	68.4 ± 7.3	70.3 ± 7.9	**72.2 ± 6.9**
Vote	**96.9 ± 3.8**	92.7 ± 5.5	91.9 ± 4.9	96.1 ± 3.8	96.8 ± 3.1	**96.9 ± 3.4**

Meta-Learning

T-based meta-learning

To create successful meta-learning through search in the model space fine granulation of methods is needed, extracting info using support features, learning from others, knowledge transfer and deep learning.

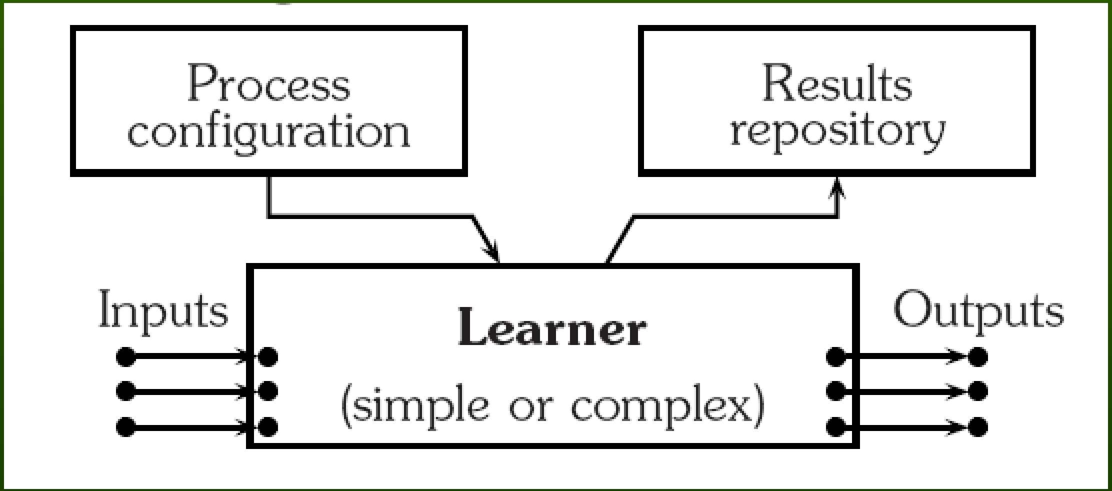
Learn to compose, using complexity guided search, various transformations (neural or processing layers), for example:

- Creation of new support features: linear, radial, cylindrical, restricted localized projections, binarized … feature selection or weighting.
- Specialized transformations in a given field: text, bio, signal analysis, ….
- Matching pursuit networks for signal decomposition, QPC index, PCA or ICA components, LDA, FDA, max. of mutual information etc.
- Transfer learning, granular computing, learning from successes: discovering interesting higher-order patterns created by initial models of the data.
- Stacked models: learning from the failures of other methods.
- Schemes constraining search, learning from the history of previous runs at the meta-level.

Real meta-learning!

Search space of all possible models is too large to explore it exhaustively, design system architecture to support knowledge-based search.

Meta-learning: learning how to learn, replace experts who search for best models making a lot of experiments.

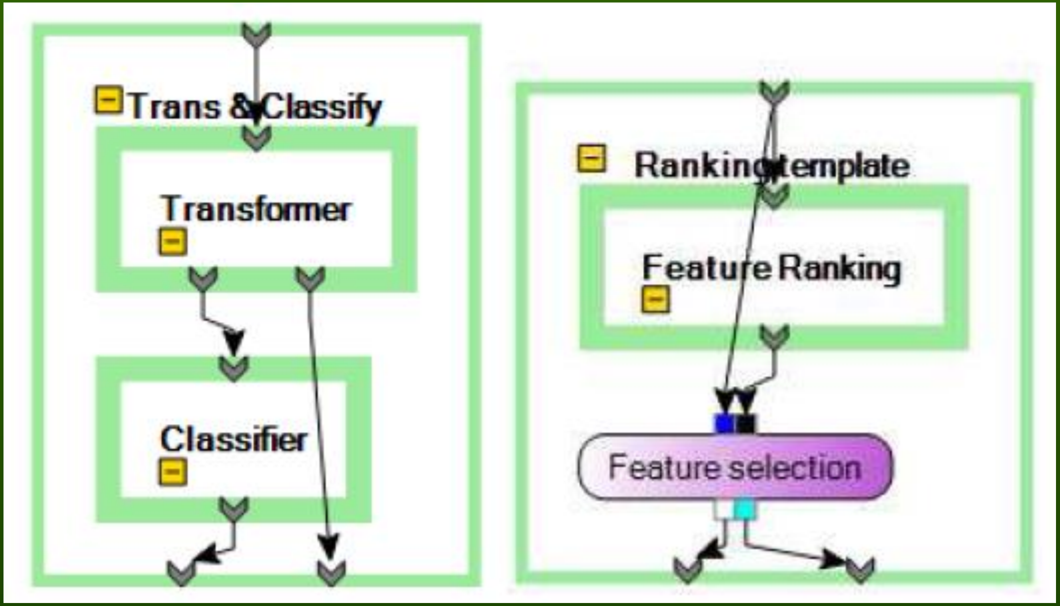


- Abstract view, uniform I/O, uniform results management.
- Directed acyclic graphs (DAG) of boxes representing scheme
- placeholders and particular models, interconnected through I/O.
- Configuration level for meta-schemes, expanded at runtime level.

An exercise in software engineering for data mining!

Intemi, Intelligent Miner

Meta-schemes: templates with placeholders.



- May be nested; the role decided by the input/output types.
- Machine learning generators based on meta-schemes.
- Granulation level allows to create novel methods.
- Complexity control: Length of the program/errors + log(time)
- A unified meta-parameters description, defining the range of sensible values and the type of the parameter changes.

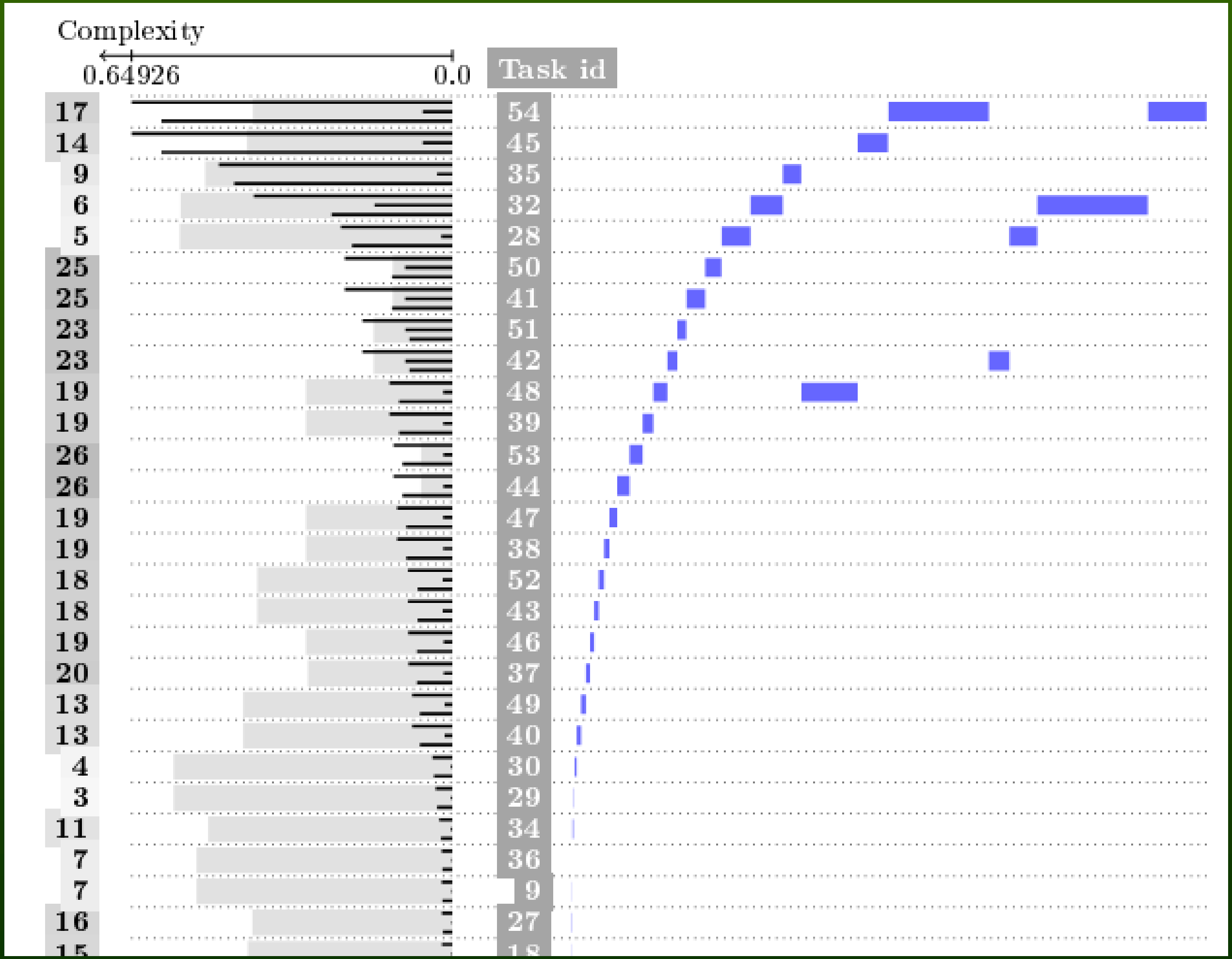
Complex machines on vowel data

Number on far left = final ranking.

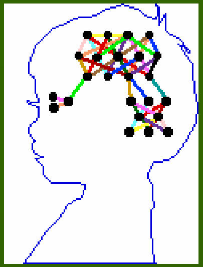
Gray bar = accuracy

Small bars (up-down) show estimation of: total complexity, time, memory.

Numbers in the middle = process id (refer to models in the previous table).



Summary



1. Challenging data cannot be handled with existing DM tools.
2. Visualization of hidden neuron's shows that frequently perfect but non-separable solutions are found despite base-rate outputs.
3. Linear separability is not the best goal of learning, other targets that allow for easy handling of final non-linearities may work better.
4. *k*-separability defines complexity classes for non-separable data.
5. Similarity-based framework enables meta-learning as search in the model space, heterogeneous systems add fine granularity.
6. Transformation-based learning shows the need for component-based approach to DM, discovery of simplest models and support features.
7. SFM and ULM may do more than SVMs.
8. Meta-learning should replace experts in automatically creating new optimal learning methods on demand.

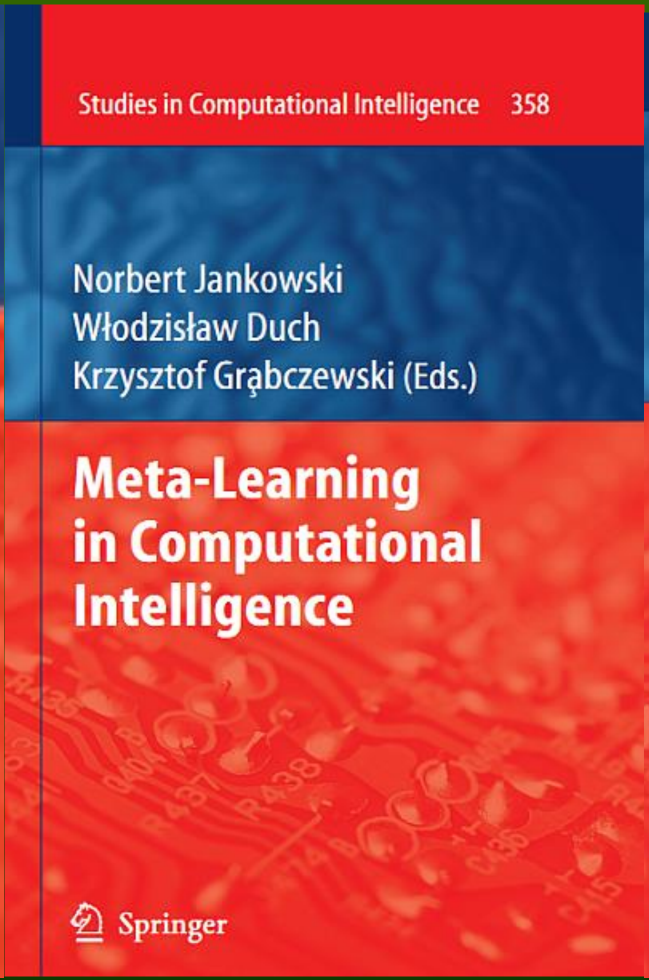
Many things to finish … Can deep learning do the same?

Studies in Computational Intelligence 498

Krzysztof Grąbczewski

Meta-Learning in Decision Tree Induction

Springer


Studies in Computational Intelligence 358

Norbert Jankowski
Włodzisław Duch
Krzysztof Grąbczewski (Eds.)

Meta-Learning in Computational Intelligence

Springer

Studies in Computational Intelligence 63

Włodzisław Duch
Jacek Mańdziuk (Eds.)

Challenges for Computational Intelligence

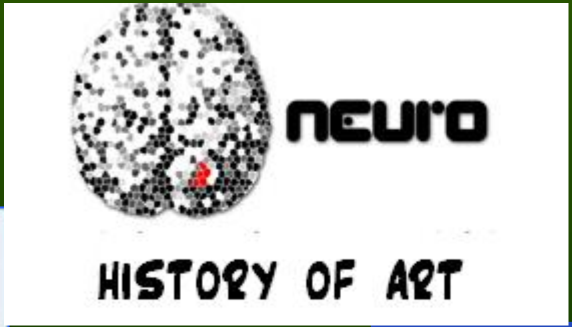
Springer

Soul or brain: what makes us human? Interdisciplinary Workshop with theologians, Toruń 19-21.10.2016

Monthly international developmental seminars (2017): Infants, learning, and cognitive development

Disorders of consciousness 17-21.09.2017

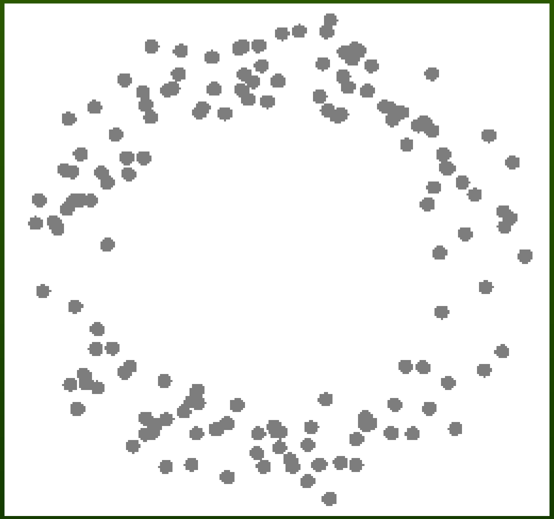
Autism: science, therapies 23.05.2017

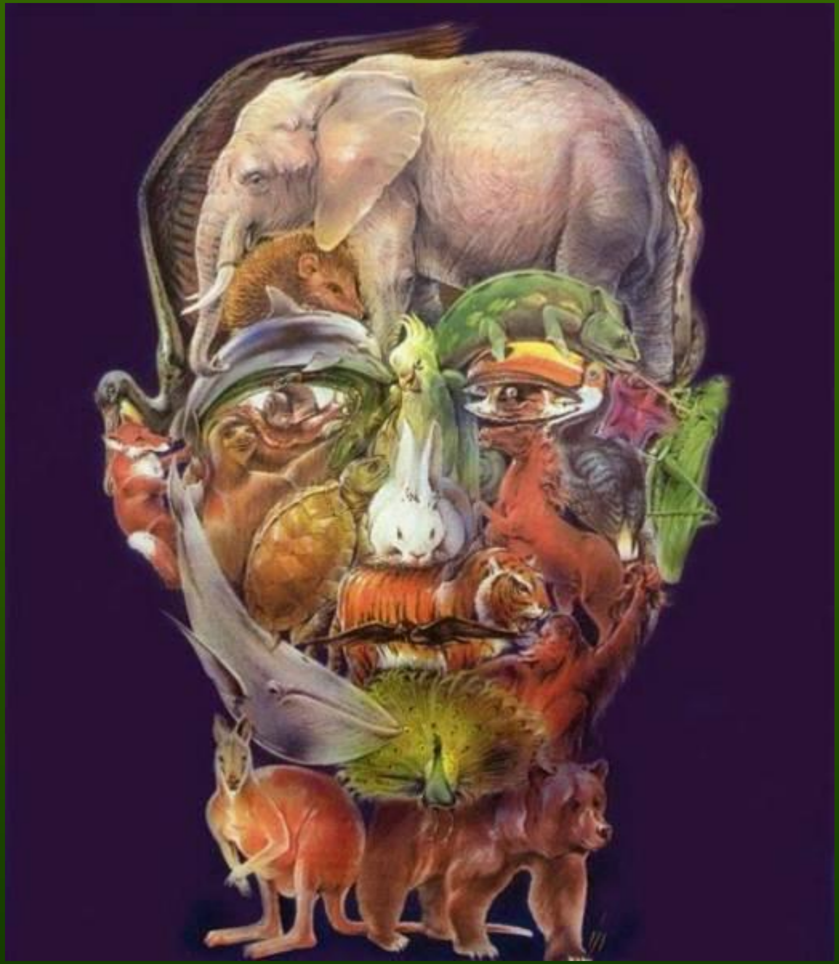


Cognitivist Autumn in Toruń 2011
PHANTOMOLOGY:
the virtual reality of the body
2011 Torun, Poland

Cognitivist Autumn in Toruń 2010
MIRROR NEURONS:
from action to empathy
April , 14-16 2010 Torun, Poland

Thank for synchronization of your neurons





Google: W. Duch

=> talks, papers, projects, lectures …