Neural Network Theory and Applications: Lecture Three

Bao-Liang Lu（吕宝粮）
Dept. of Computer Science & Engineering
Shanghai Jiao Tong University
Office: 3-431 Dianxin Bld, Minhang
Tel: 3420-5422
bllu@sjtu.edu.cn

Outline of Lecture Three

- Summary of MLP and BP algorithm
- Multi-sieving Network Model
- Min-max modular (M3) Network
- Solving Imbalance problem
- Min-Max Modular Network with Gaussian-Zero-Cross (GZC) Function
- Radial Base Function Networks

Summary of MLP and BP Algorithm

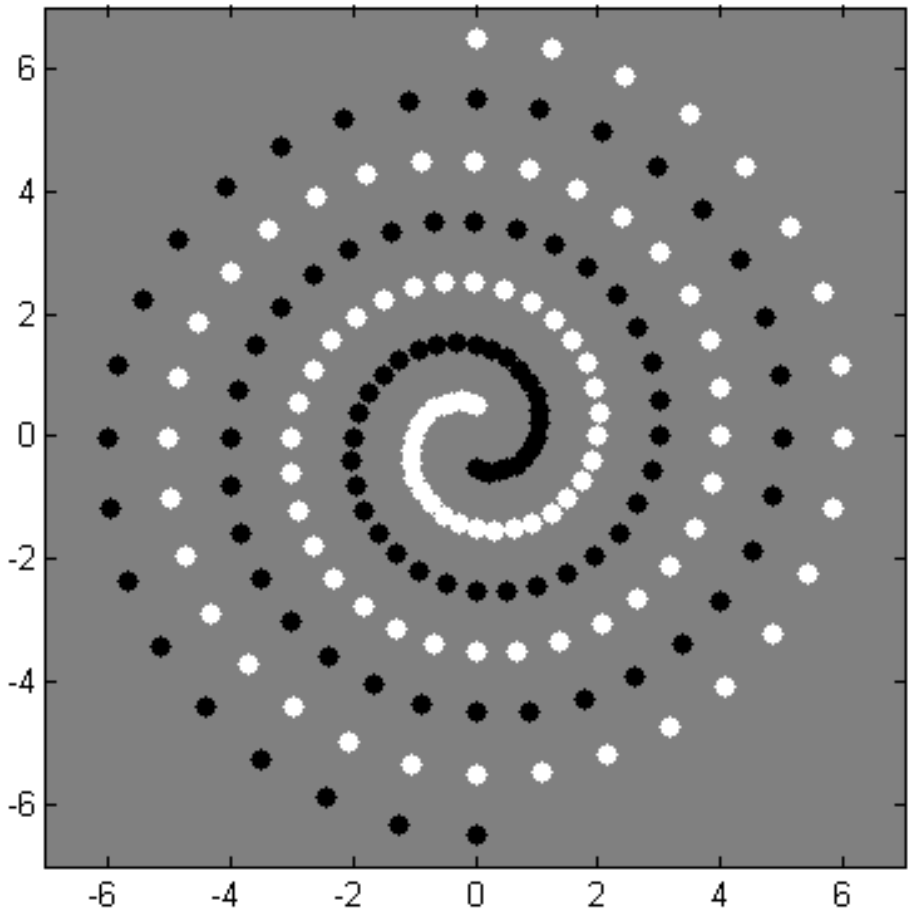
- Universal approximation capability
- Can solve linear non-separable problem
- Two modes of training
  - sequential; online; stochastic
  - batch; mini-batch
- Three factors affect generalization:
  - The size and representativeness of the training set
  - The architecture of the neural network
  - The physical complexity of the problem at hand
- How to select the number of hidden units?
- BP algorithm converges very slowly!

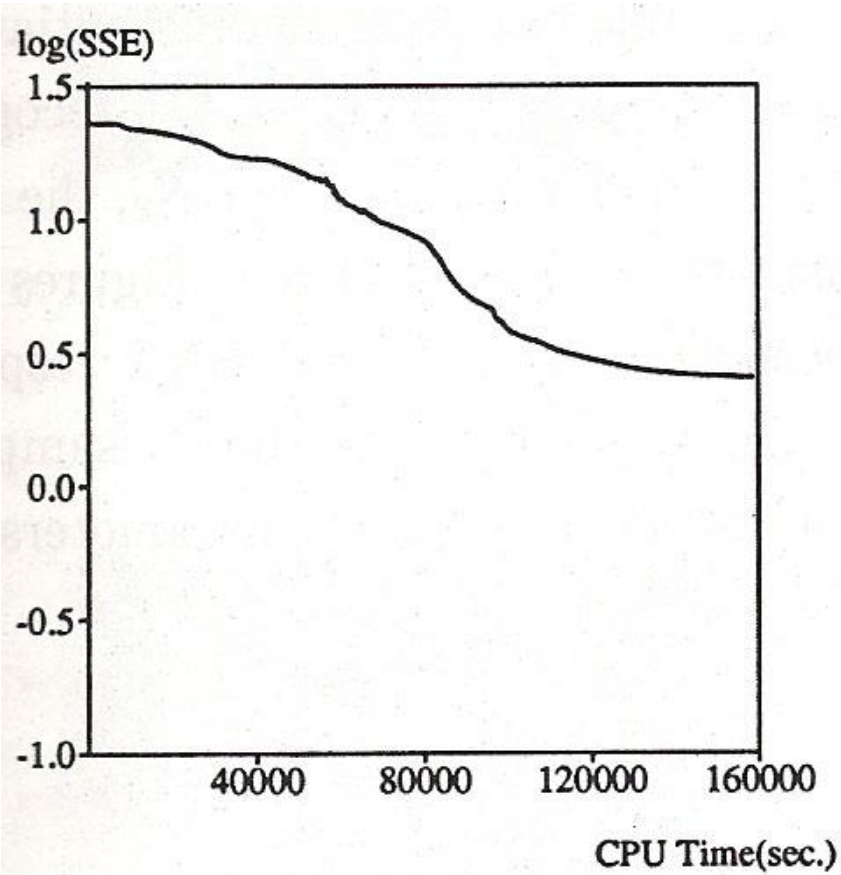
Two ways of determining network size

- Pruning
- Growing

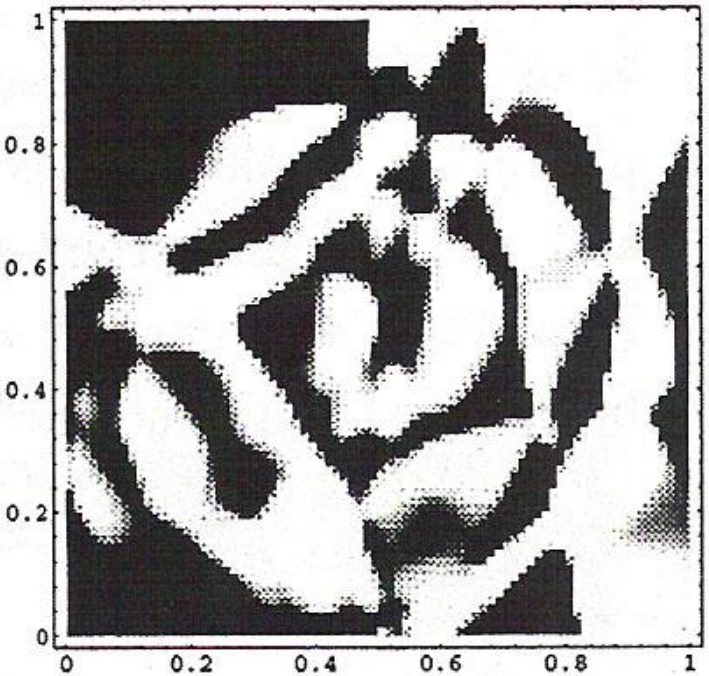
Multi-Sieving Neural Network (Lu, et al., 1994)

Two Spirals Problem



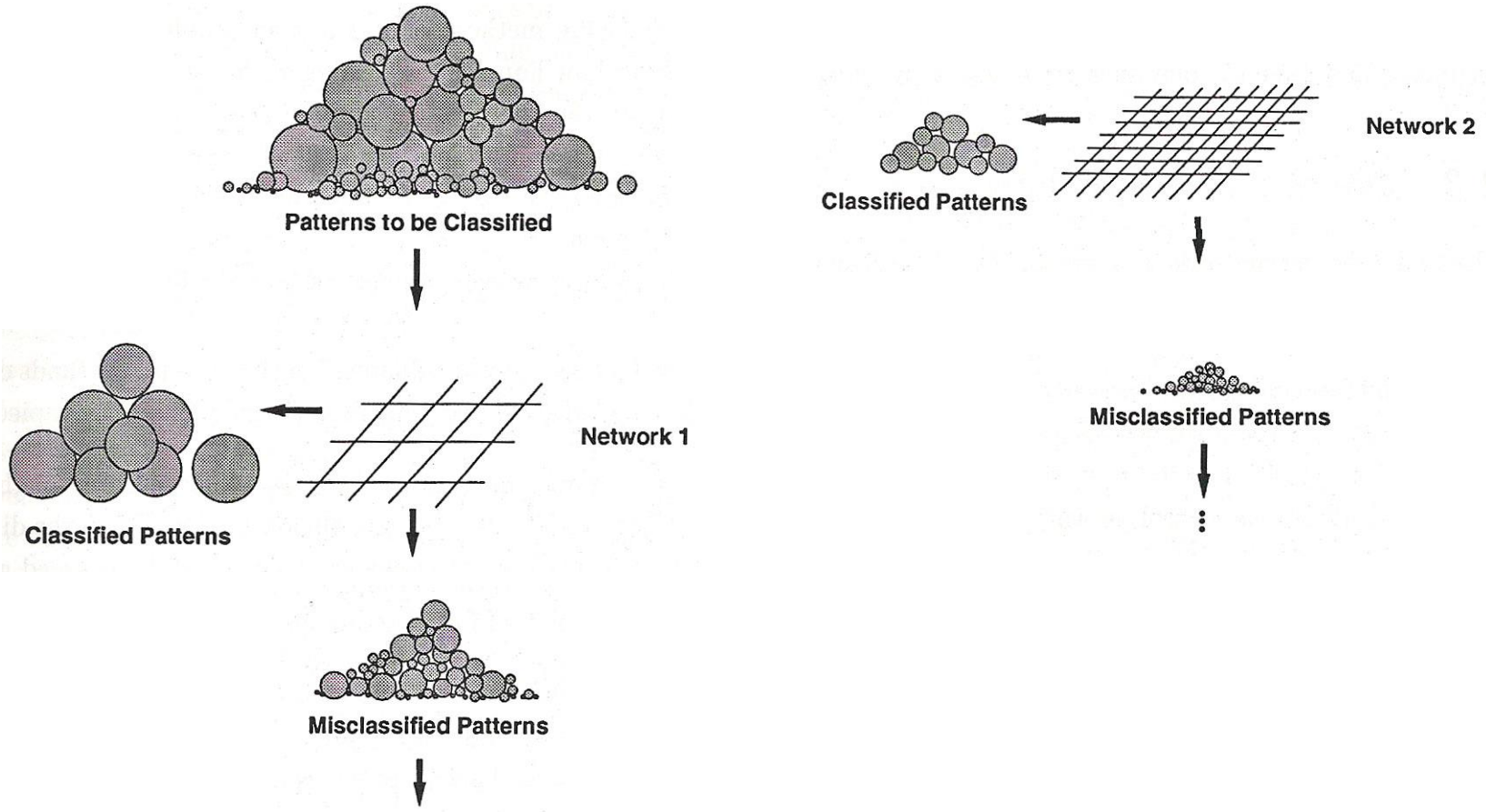

log(SSE)
1.5
1.0
0.5
0.0
-0.5
-1.0
40000
80000
120000
160000
CPU Time(sec.)

(a)


1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1

(b)

Multi-Sieving Network Model



Structure of the Multi-Sieving Network

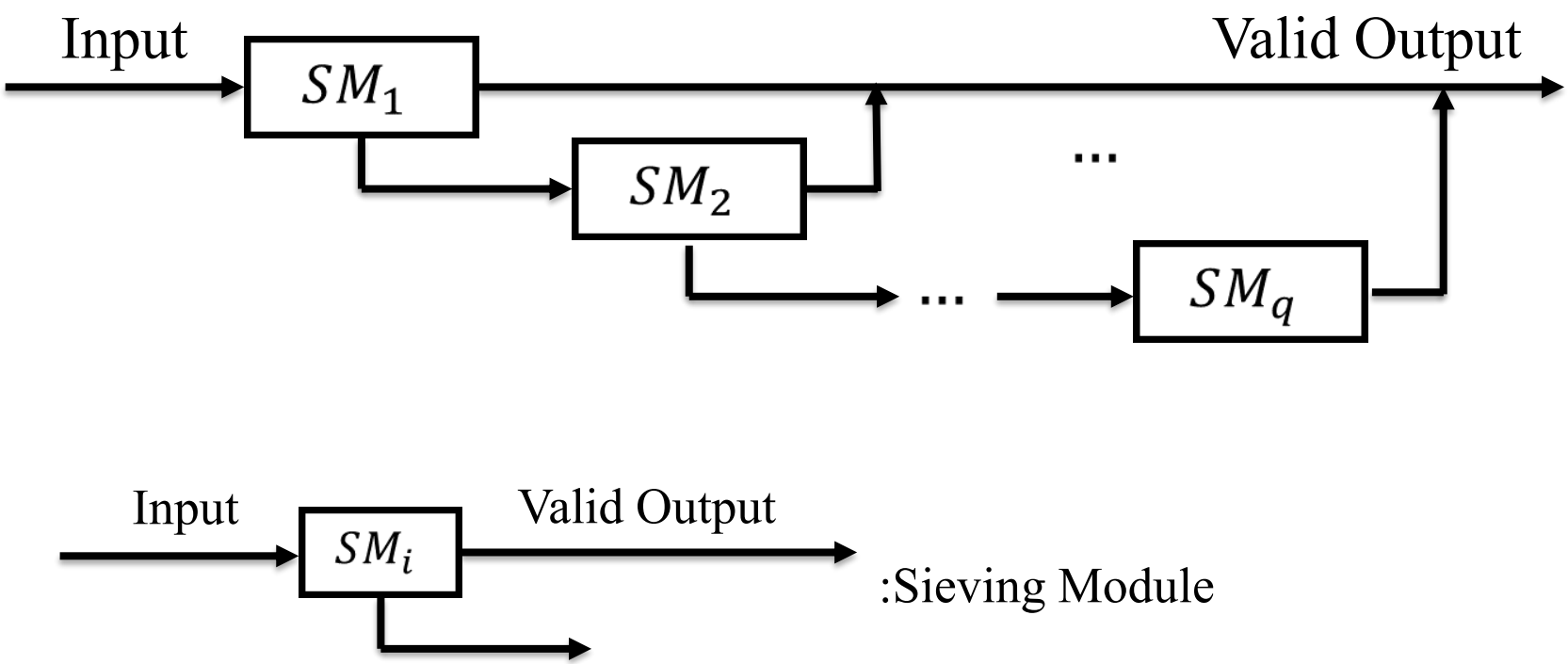


Figure 4.2: The structure of the multi-sieving neural network

Structure of the Multi-Sieving Network (2)

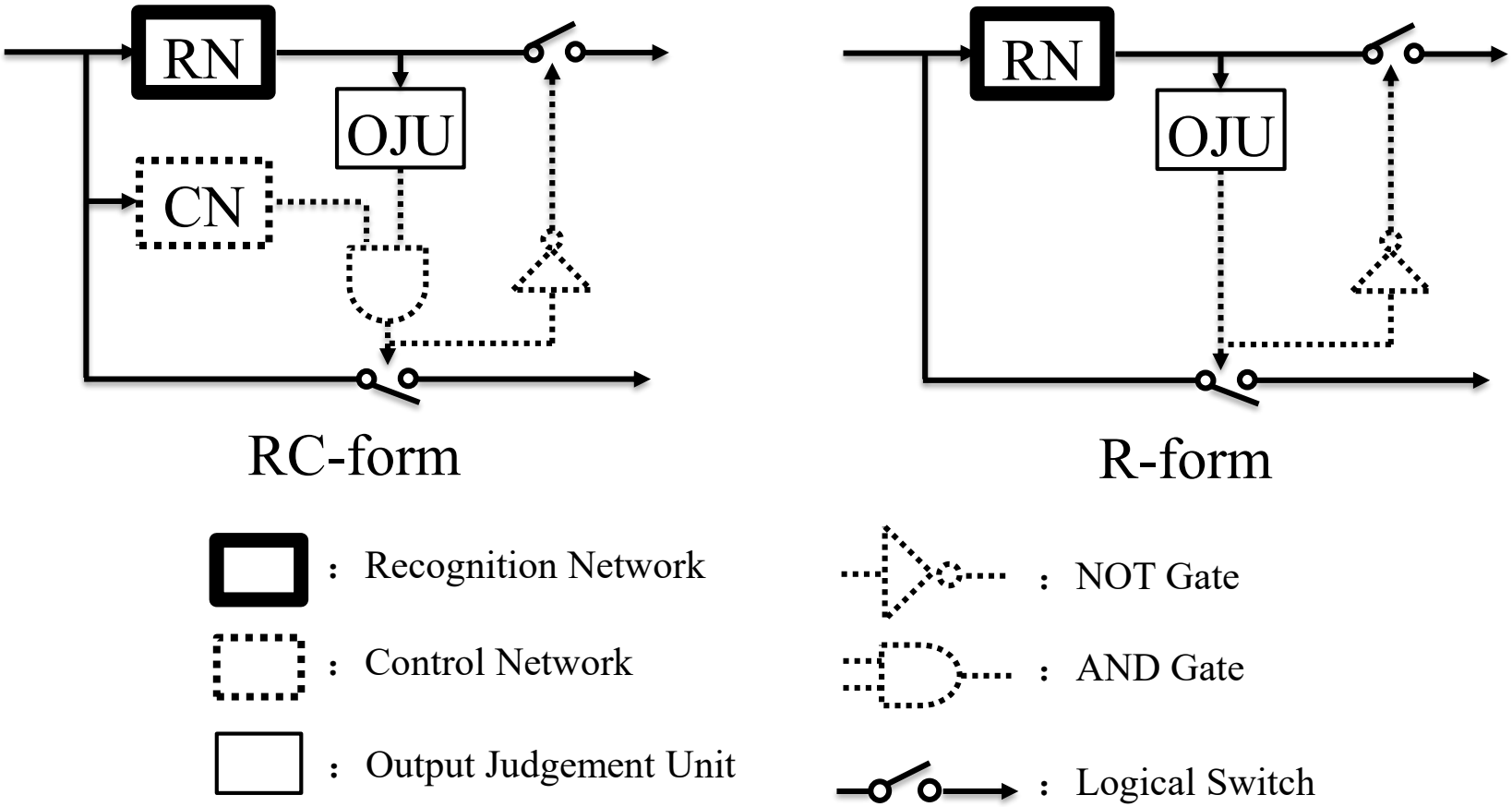


Figure 4.3: Two forms of sieving modules

Valid, Pseudo Valid, and Invalid Outputs

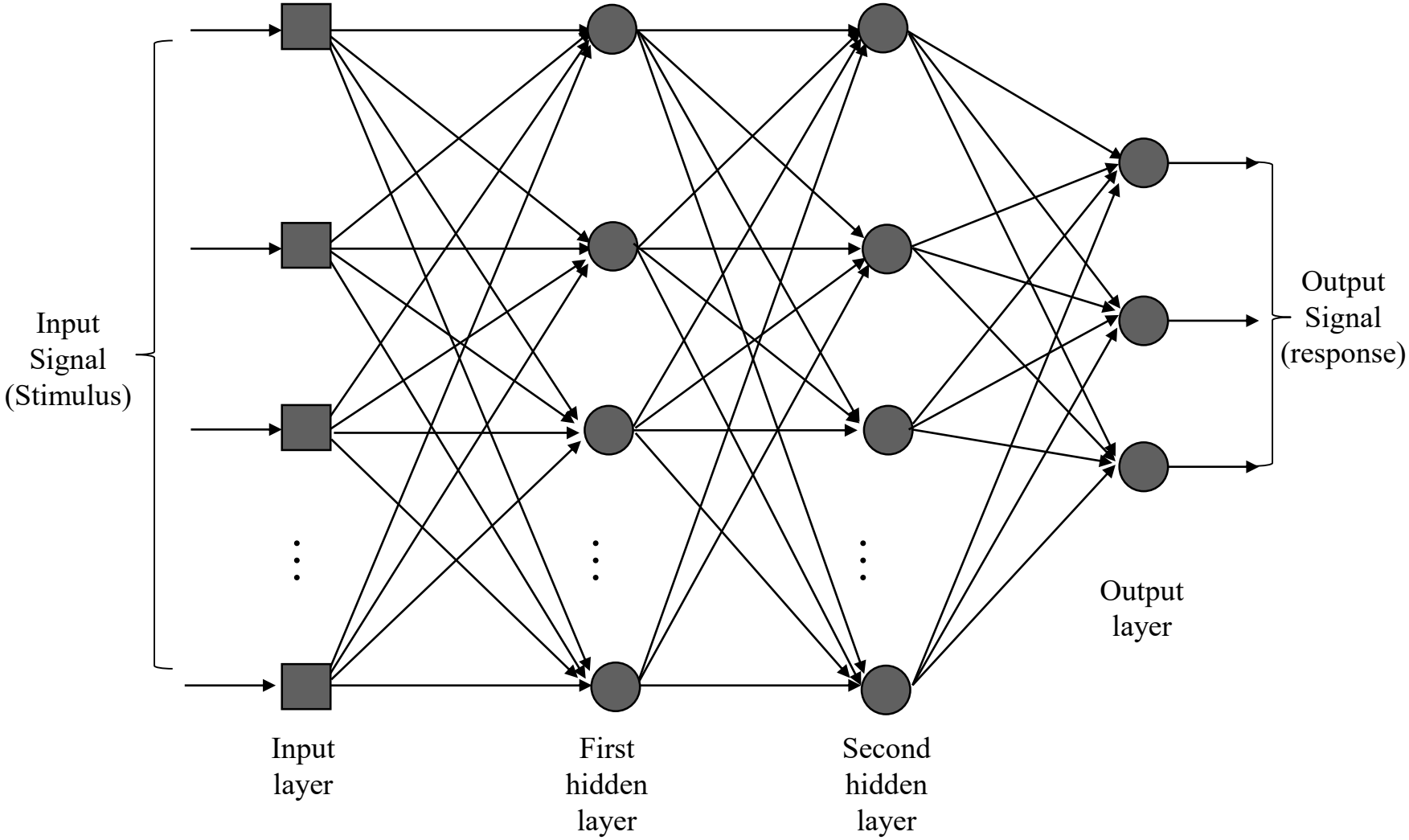
For classification task of $p+1$ sets, we use p or $p+1$ output units. For the kth recognition network RN_k, a desired output pattern $\bar{\boldsymbol{x}}_i^O = \{\bar{x}_{i1}^O, \bar{x}_{i2}^O, \dots, \bar{x}_{iN_k}^O\}$ must satisfy one of the following rules:

(a) $\forall j(\bar{x}_{ij}^O < x_{low}^O)$ for $j \in B_k$ or

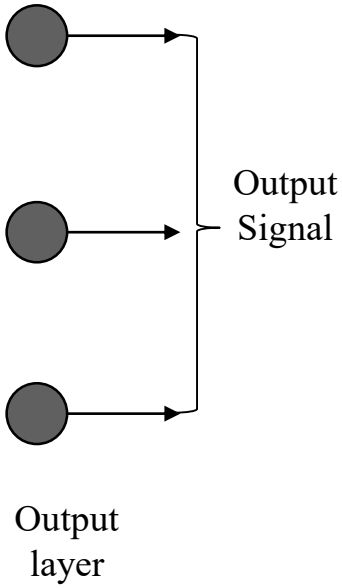
(b) $\exists j\left(\bar{x}_{ij}^O > x_{high}^O\right) \wedge \forall l_{l \neq j}(\bar{x}_{il}^O < x_{low}^O)$ for j and $l \in B_k$

where $B_k = \{1, 2, \dots, N_k\}$, N_k is the number of output units in RN_k, and x_{low}^O and x_{high}^O represent the low and high bounds for the outputs, respectively. For example, three output units can only represent four valid outputs as follows: (0,0,0), (0,0,1), (0,1,0) and (1,0,0). Other four binary codes, (0,1,1), (1,0,1), (1, 1,0) and (1, 1, 1), are considered to be invalid.

Representation of Input and Output



Representation of Desired Output



Class 1	0	0	0
Class 2	0	0	1
Class 3	0	1	0
Class 4	1	0	0

Binary Representation: 2^N

Valid, Pseudo Valid, and Invalid Outputs (2)

For a given input pattern $\bar{\boldsymbol{x}}_i^I$, RN_k may generate three kinds of actual outputs:

(a). *Valid output*: The valid output is the correct output generated by the recognition network. That is,

$$\forall j |\bar{x}_{ij}^O - x_{ij}^{RN_k}| < \delta \quad \text{for} \quad j \in B_k \qquad (4.1)$$

where $\bar{x}_{ij}^O$ is the desired output of the jth unit, $x_{ij}^{RN_k}$ is the actual output of the jth output unit of the kth recognition network RN_k, and δ denotes a tolerance. If the desired values of the output units are set to 0 and 1, then, $x_{low}^O = \delta$ and $x_{high}^O = 1 - \delta$.

Valid, Pseudo Valid, and Invalid Outputs (3)

(b). *Pseudo valid output*: The recognition network may generate an output which follows the coding rule mentioned above, but is not correct. We called such an output a *pseudo valid output*:

$$\exists j_{j\neq h}\left(x_{ij}^{RN_k} > x_{high}^{O}\right) \wedge \forall l_{l\neq j}\left(x_{il}^{RN_k} < x_{low}^{O}\right) \vee \forall l\left(x_{il}^{RN_k} < x_{low}^{O}\right)$$

$$\text{for } j \text{ and } l \in B_k \qquad (4.2)$$

where the desired output of the hth unit satisfies $\bar{x}_{ih}^{O} > x_{high}^{O}$.

(c). *Invalid output*: Otherwise.

For example, if the desired output pattern is (0, 0, 1), $\delta = 0.2$, $x_{high}^{O} = 0.8$, and $x_{low}^{O} = 0.2$, then, (0.1, 0.1, 0.9) is a valid output, (0.9, 0.1, 0.1) and (0.1, 0.1, 0.1) are two pseudo valid outputs, and (0.9, 0.1, 0.9) is an invalid output.

(a) The output judgement unit is used to differentiate the invalid output from other two kinds of outputs. OJU generates 1 or 0 according to

$$O_{OJU,k} = \begin{cases} 1, \text{ if } x_{ij}^{RN_k} \text{ is a valid or pseudo valid output;} \\ 0, \text{ otherwise} \end{cases}$$

where $O_{OJU,k}$ is the output of OJU in the kth sieving module, $A = \{1, 2, \ldots, t_k\}$, t_k is the number of examples used for training the kth recognition network.

(b) The control network is used to differentiate the valid output from the pseudo valid output. Its output is also 1 or 0, which is determined by

$$O_{CN,k} = \begin{cases} 1, \text{ if } x_{ij}^{RN_k} \text{ is a valid output;} \\ 0, \text{ otherwise} \end{cases}$$

where $O_{CN,k}$ is the output of the control network in the kth sieving module.

Multi-Sieving Learning Algorithm

Let T_1 be a set of t_1 training samples to be learned:

$$T_1 = \{(\overline{\boldsymbol{x}}_i^I, \overline{\boldsymbol{x}}_i^O) | for\ i = 1, 2, \ldots, t_1\}$$

where $\overline{\boldsymbol{x}}_i^I \in \boldsymbol{R}^{N_I}$ and $\overline{\boldsymbol{x}}_i^O \in \boldsymbol{R}^{N_O}$ are the input and the desired output of the ith sample, respectively.

The multi-sieving learning algorithm works as follows:

Step 1 : Initially, one recognition network, namely RN_1, is trained on the original set T_1 up to a certain number of epochs. Let $m = 1$,and proceed to the following steps.

Step 2 : Compute the number of valid outputs, $N_{VO,m}$, and the number of pseudo valid outputs, $N_{PVO,m}$, according to Eqs. 4.1 and 4.2, respectively.

Multi-Sieving Learning Algorithm (2)

Step 3 : If $\sum_{i=1}^{m} N_{VO,i} = t_1$, ie., if all t_1 samples are learned by the multi-sieving network, then the training is completed.

Step 4 : If $N_{PVO,m} = 0$, i.e., if there are no pseudo valid outputs, then the control network is unnecessary. Go to Step 6.

Step 5 : If $N_{PVO,m} > 0$, i.e., if there exist $N_{PVO,m}$ pseudo valid outputs, then, a control network, namely CN_m, is trained on the set of $N_{VO,m} + N_{PVO,m}$ samples until all of the samples are classified correctly. These samples correspond to valid or pseudo valid outputs generated by RN_m.

Multi-Sieving Learning Algorithm (3)

Step 6 : Freeze the parameters of RN_m and CN_m (if it exists), and construct the mth sieving module as shown in Fig. 4.3.

Step 7 : Remove $N_{VO,m}$ samples which have been successfully classified by RN_m from T_m and create a new set of $t_{m+1}(t_{m+1} = t_m - N_{VO,m})$ samples $T_{m+1}(T_{M+1} \subset T_m)$, which are not classified by RN_m.

Step 8 : Train RN_{m+1} on T_{m+1} up to a certain number of epochs. Let $m = m + 1$ and go back to Step 2.

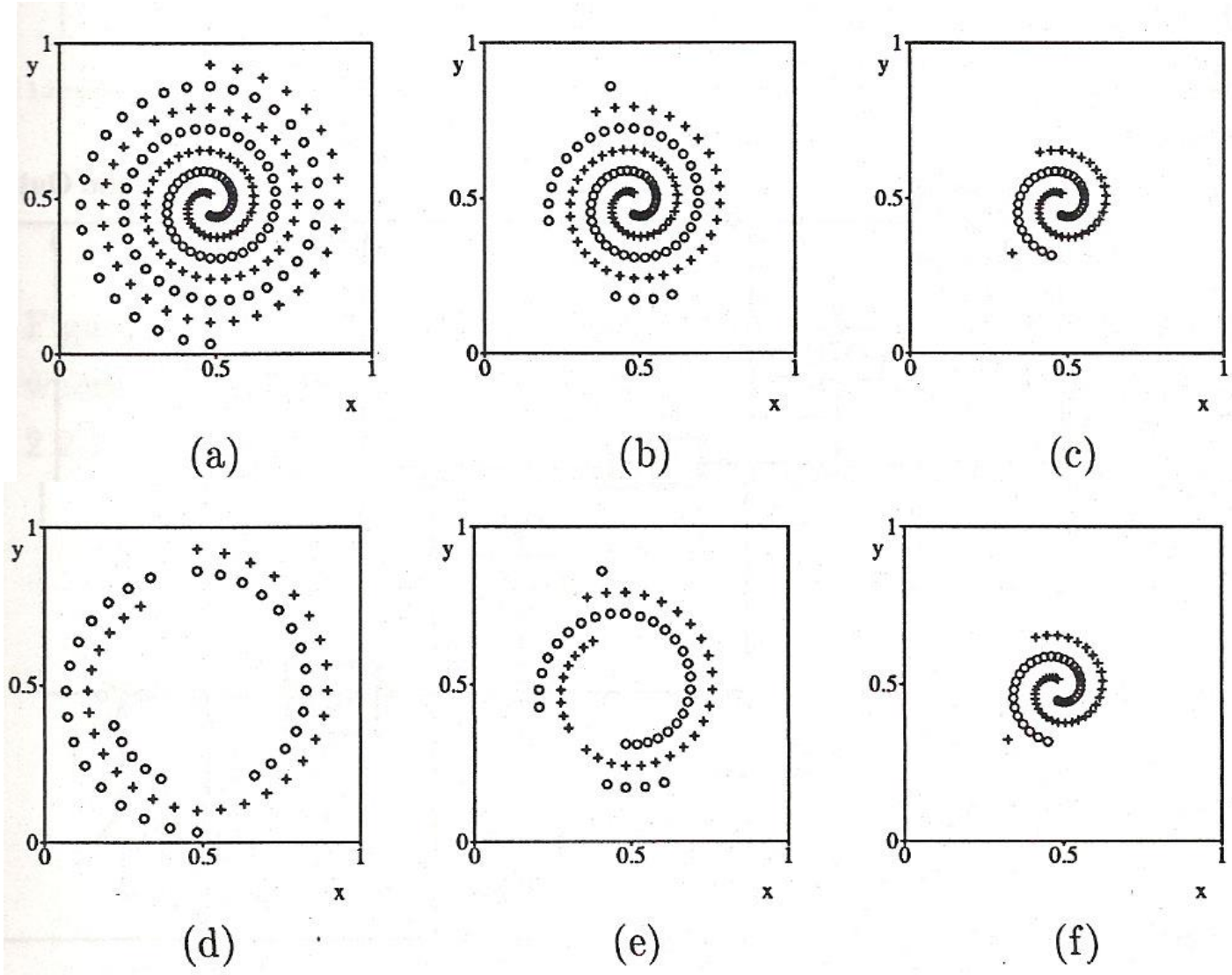


Figure 4.5: Patterns used for training RN_1 (a), RN_2 (b), and RN_3 (c). Patterns learned by the first sieving module (d), the second sieving module (e), and the third sieving module (f). For symbols "o" and "+", the network is required to generate output 0 and 1, respectively.

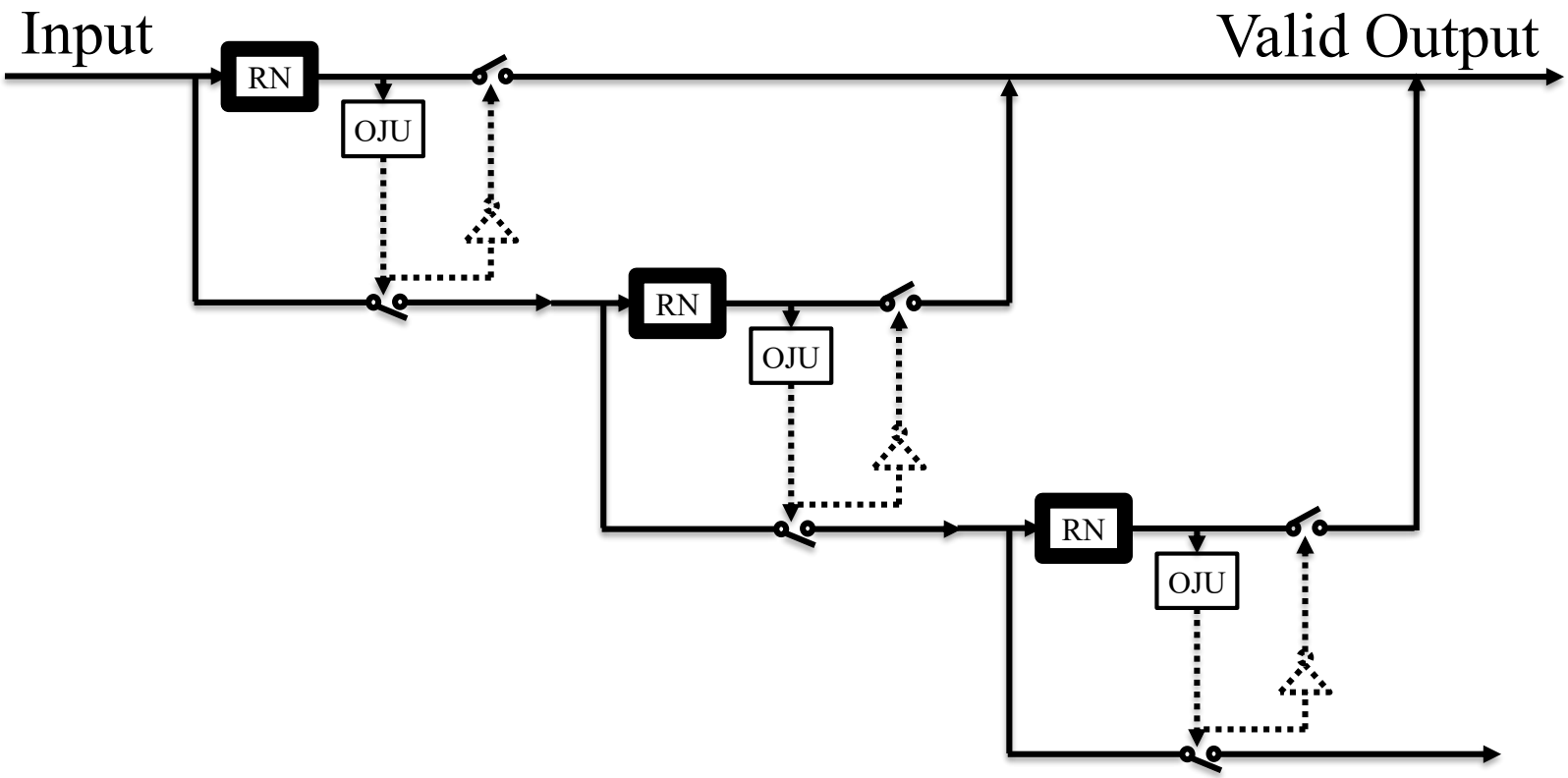
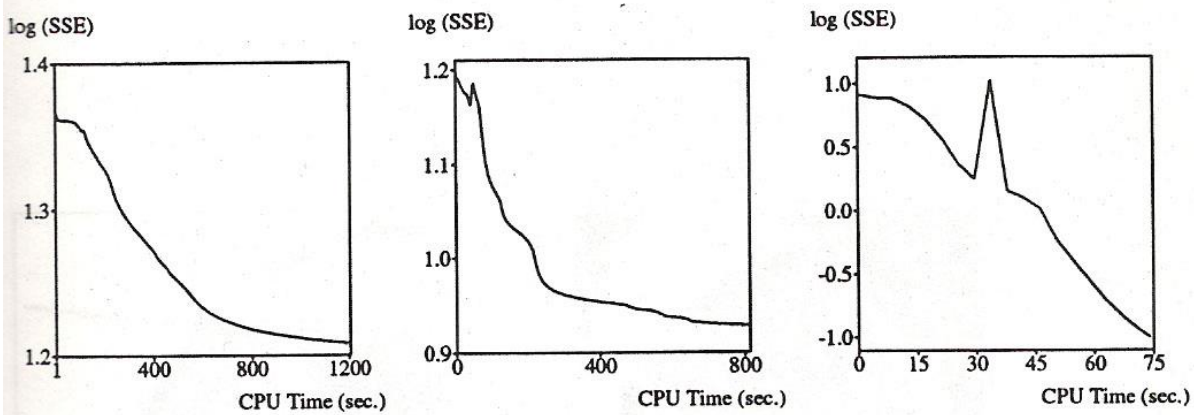


Figure 4.6: The multi-sieving network with three sieving modules for the "two-spirals" problem, namely MSN_{ts}^{3}.



(a) (b) (c)

Figure 4.7: The learning curves of RN_1 (a), RN_2 (b), and RN_3 (c), where SSE means the sum of squared error as defined in Subsection 2.2.1

Table 4.1: The learning rates for the recognition networks

	RN_1	RN_2	RN_3
η_1	0.084	0.11	0.21
η_2	0.41	0.81	0.51

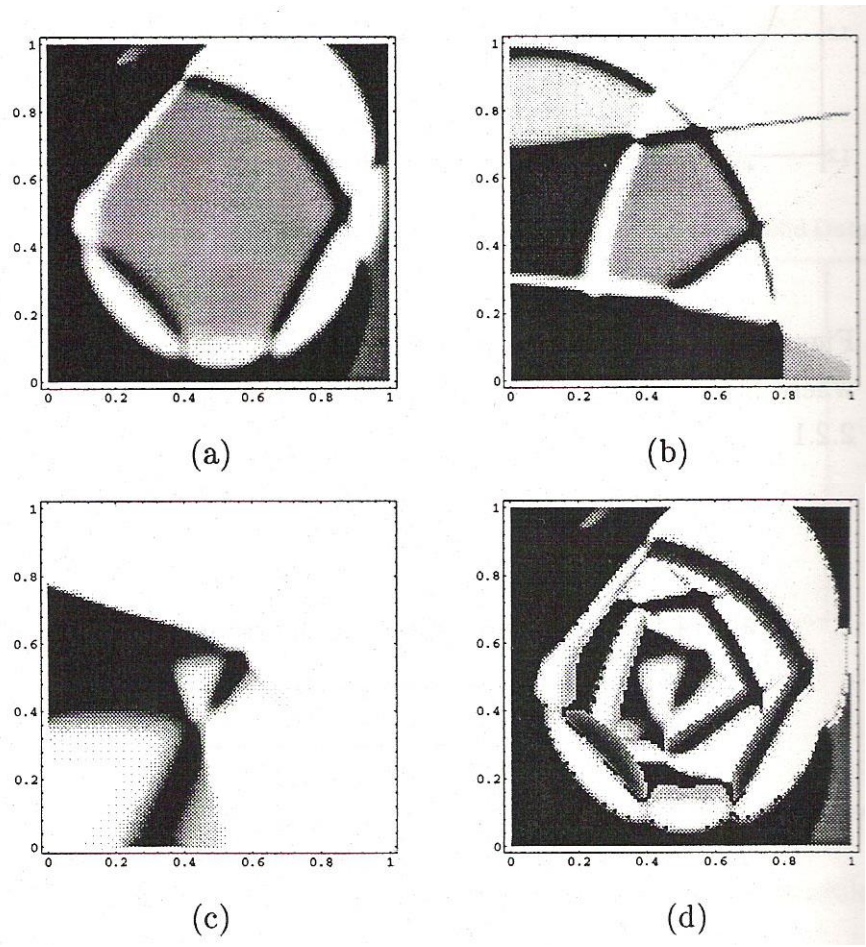


Figure 4.8: Response plots of the first sieving module (a), the second sieving module (b), the third sieving module (c), and the whole multi-sieving network (d). Black and white represent output of “0” and “1”, respectively, and grey represents intermediate value.

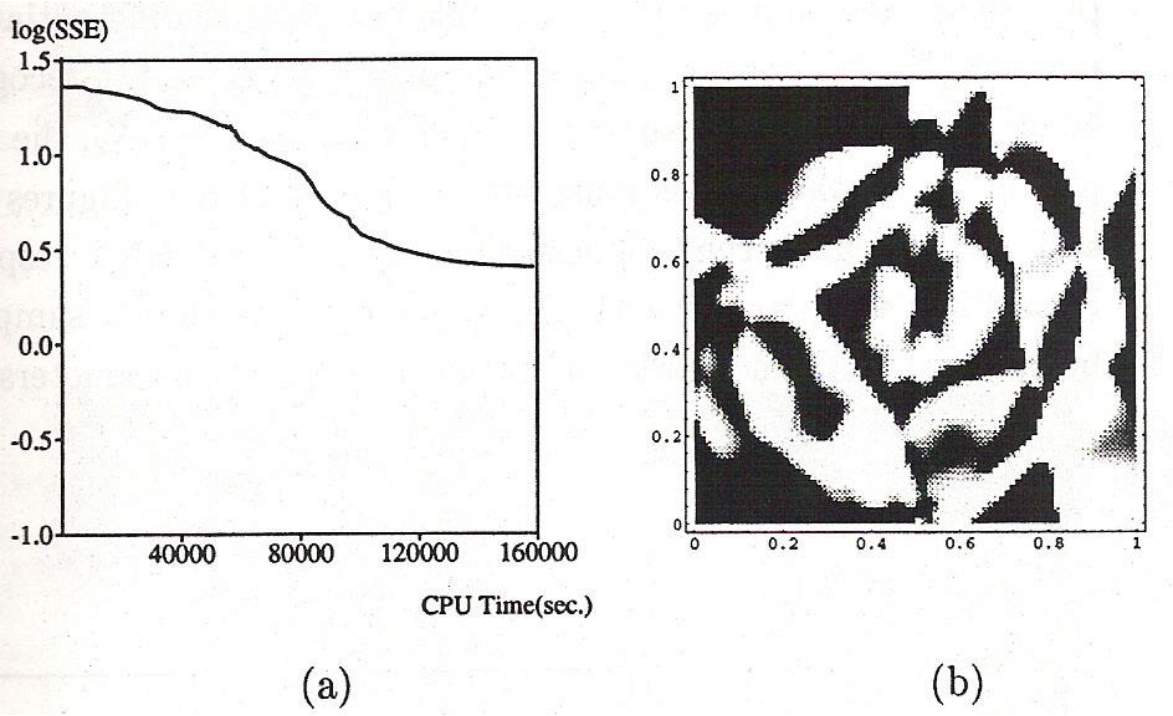
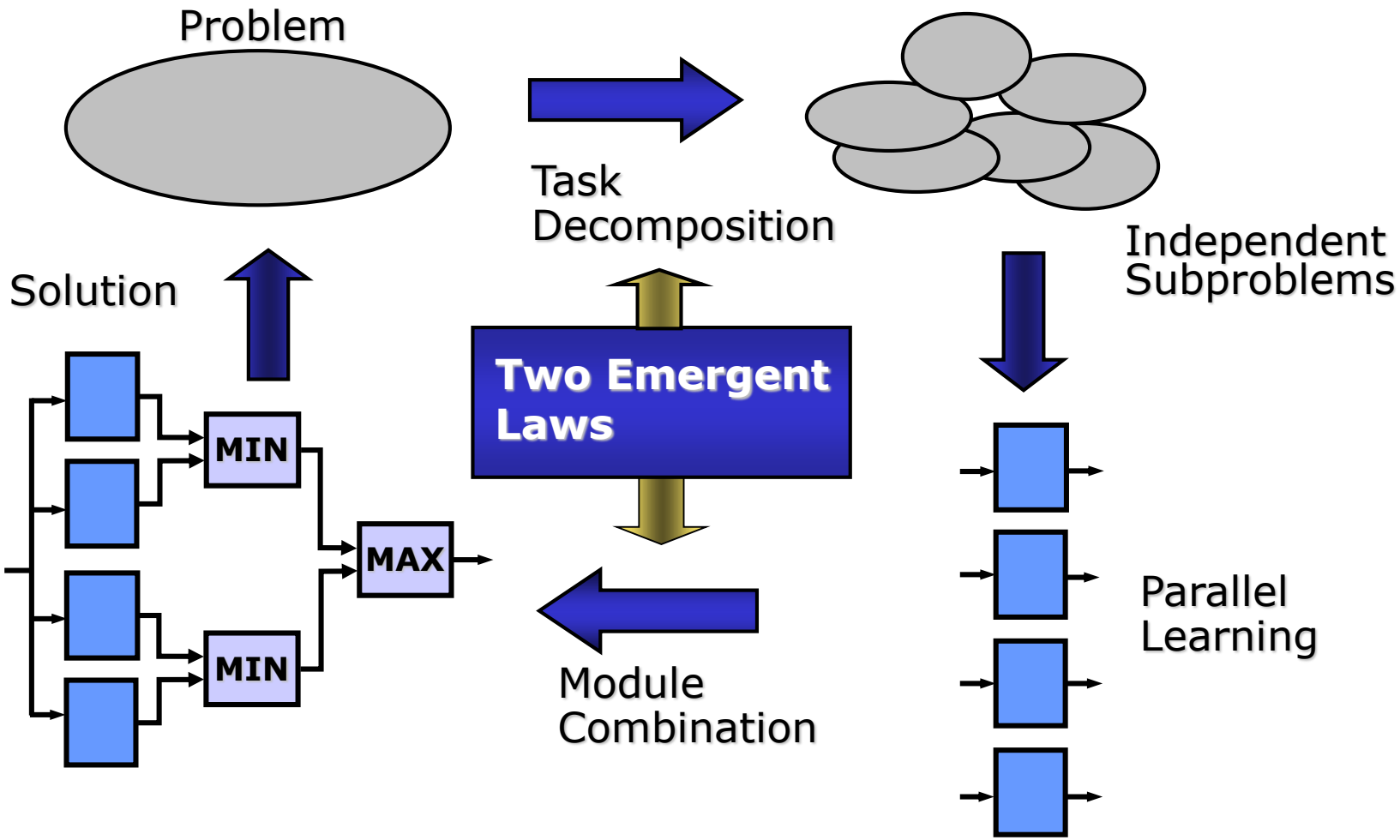


Figure 4.9: The learning curve (a) and the response plot of a plain MLQP for the “two-spirals” problem (b).

Min-Max Modular Neural Network

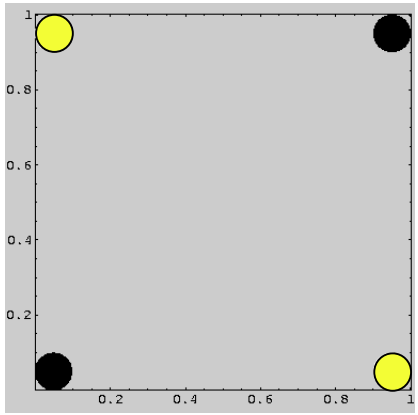
(B. L. Lu and M. Ito, 1997, 1999)

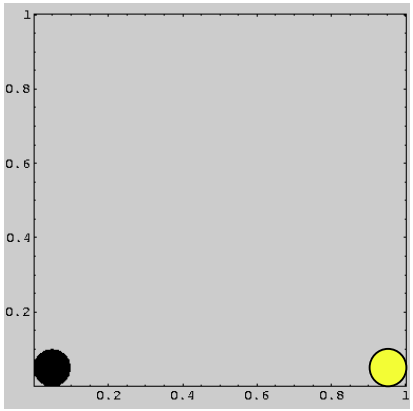
Emergent Learning in Batch Mode

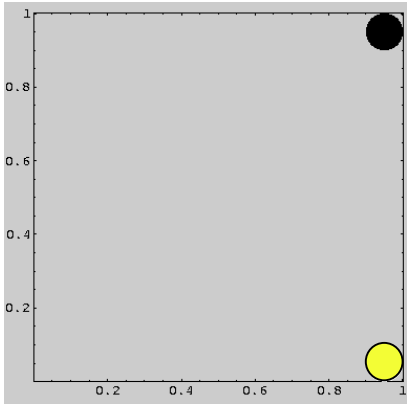


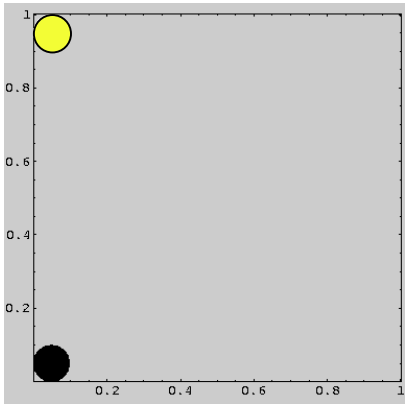
Decomposition of the XOR Problem

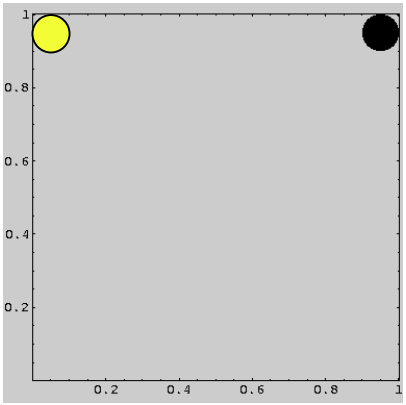
XOR Problem:







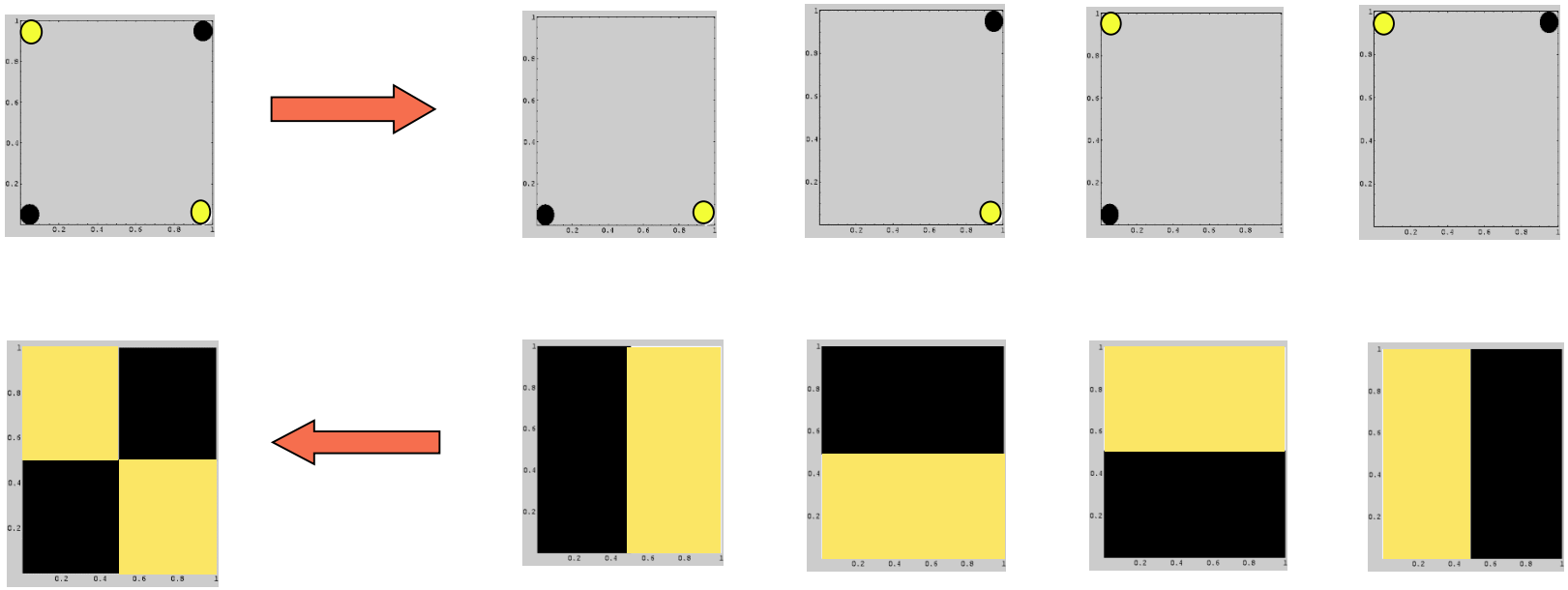




Four linearly separable problems

Decomposition *vs* Combination

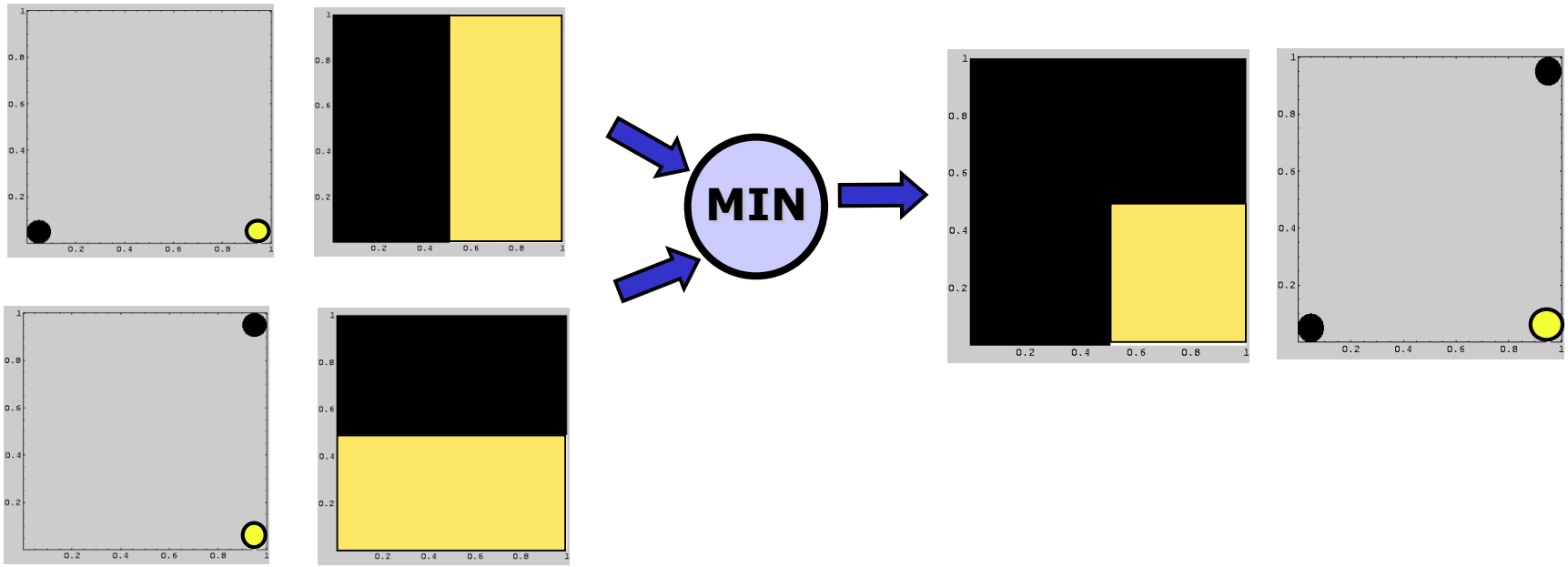
Task Decomposition: Top-Down



Module Combination: Bottom-Up

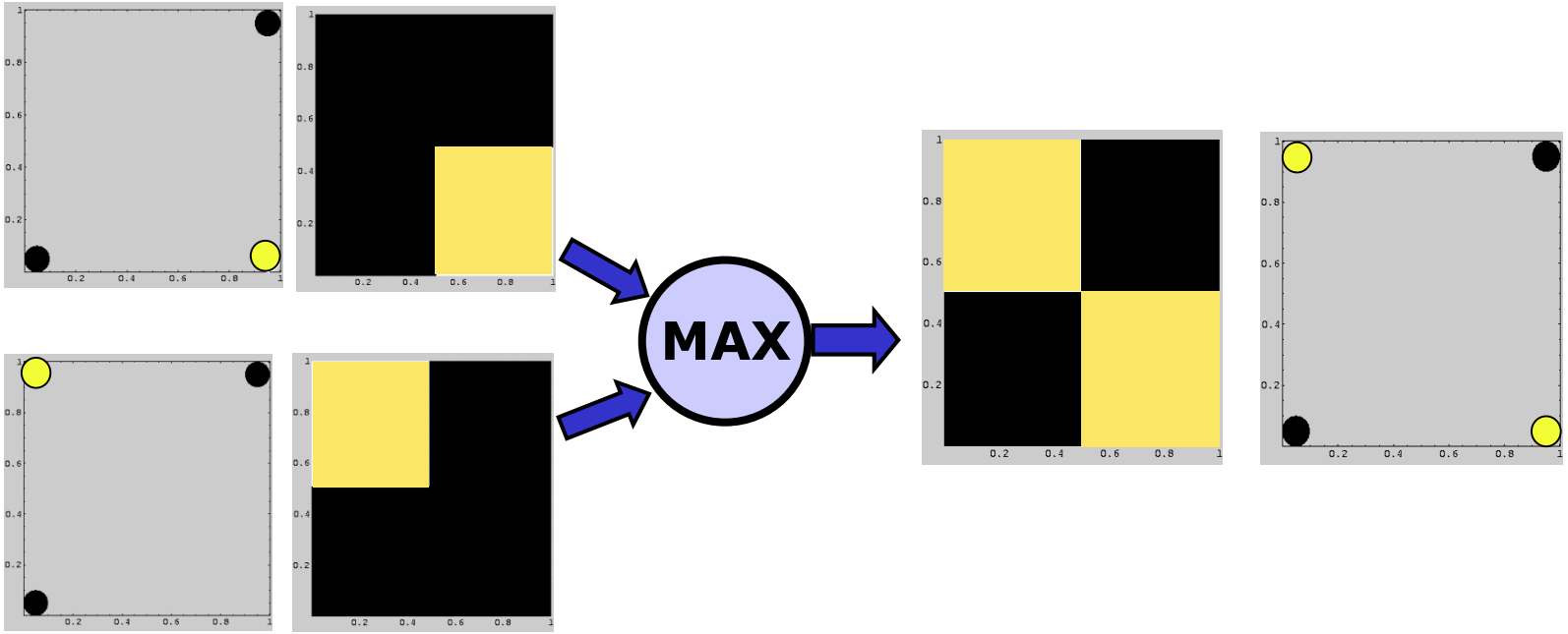
Emergent Law : Minimization

The modules, which were trained on the data sets which have the same training inputs corresponding to desired output “1”, should be integrated by the MIN unit.

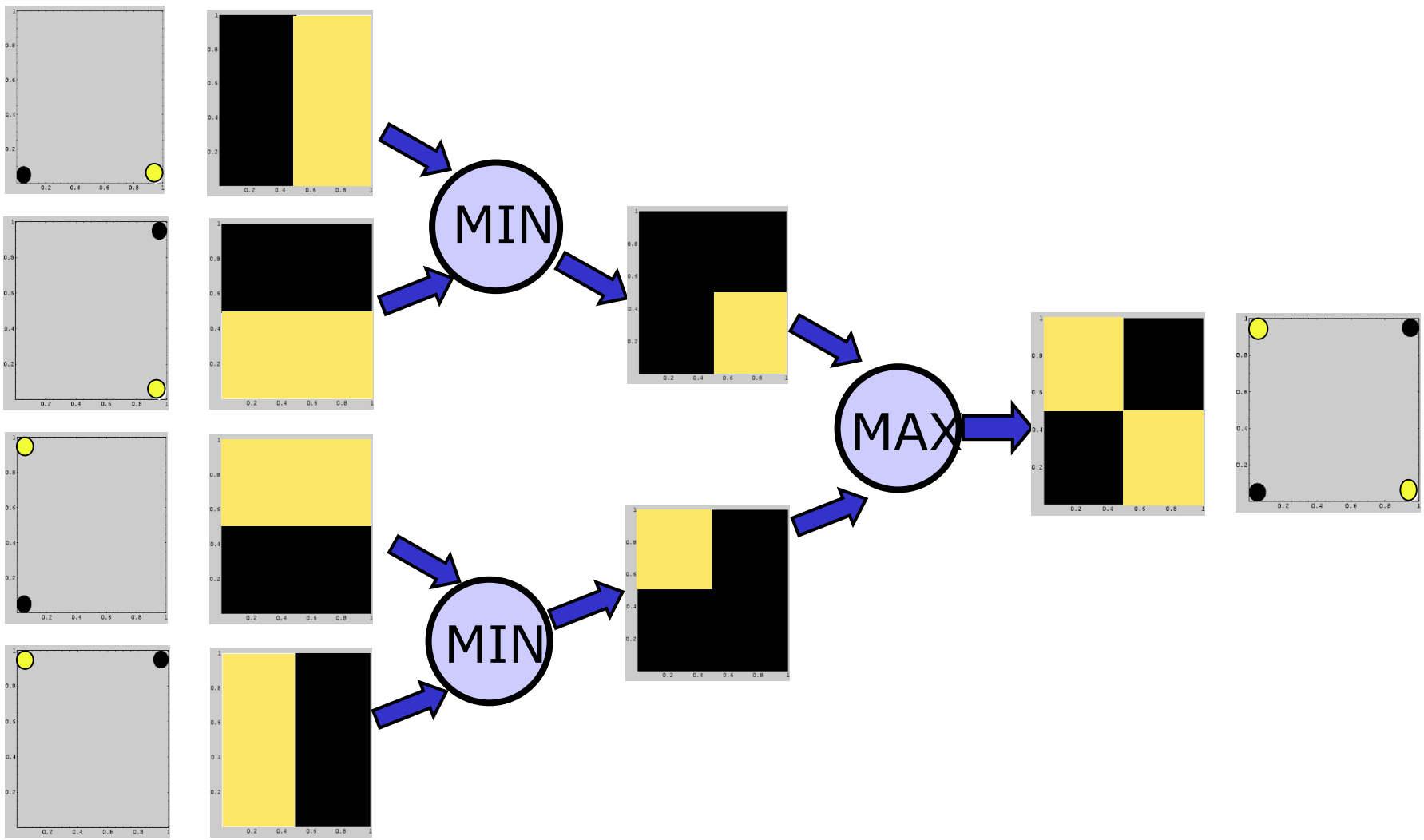


Emergent Law: Maximization

The modules, which were trained on the data sets which have the same training inputs corresponding to desired output “0”, should be integrated by the MAX unit.

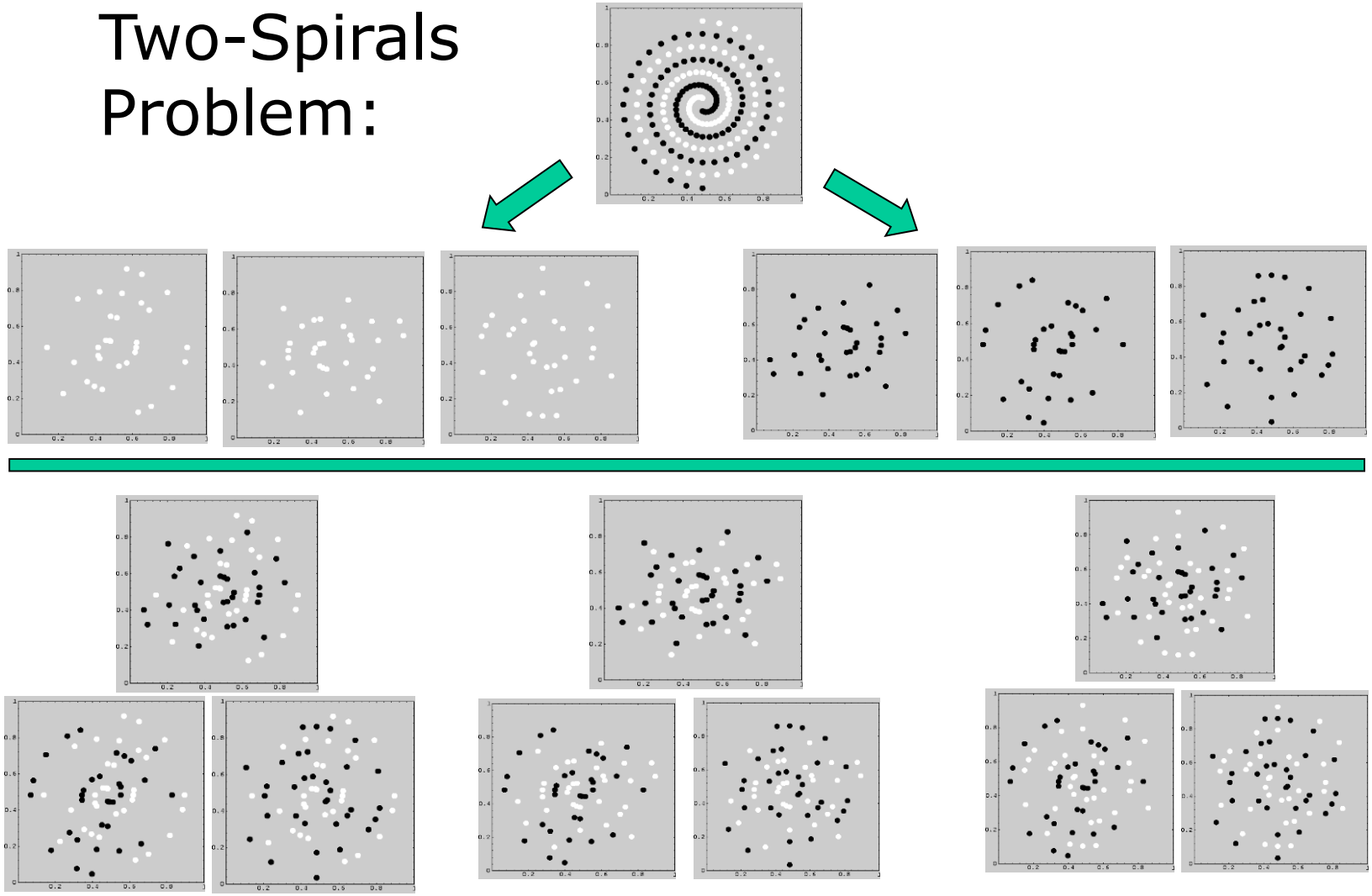


Emergent Learning of XOR problem

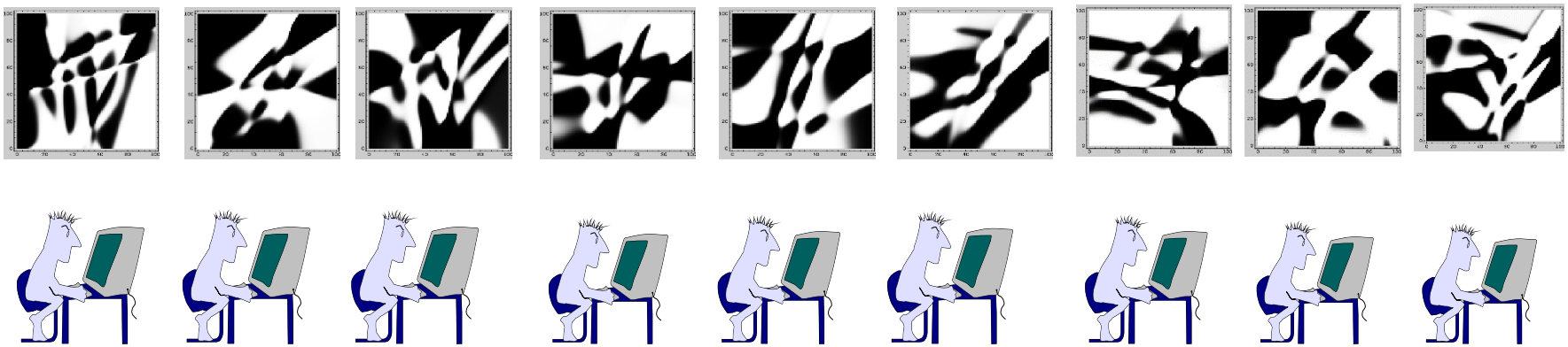


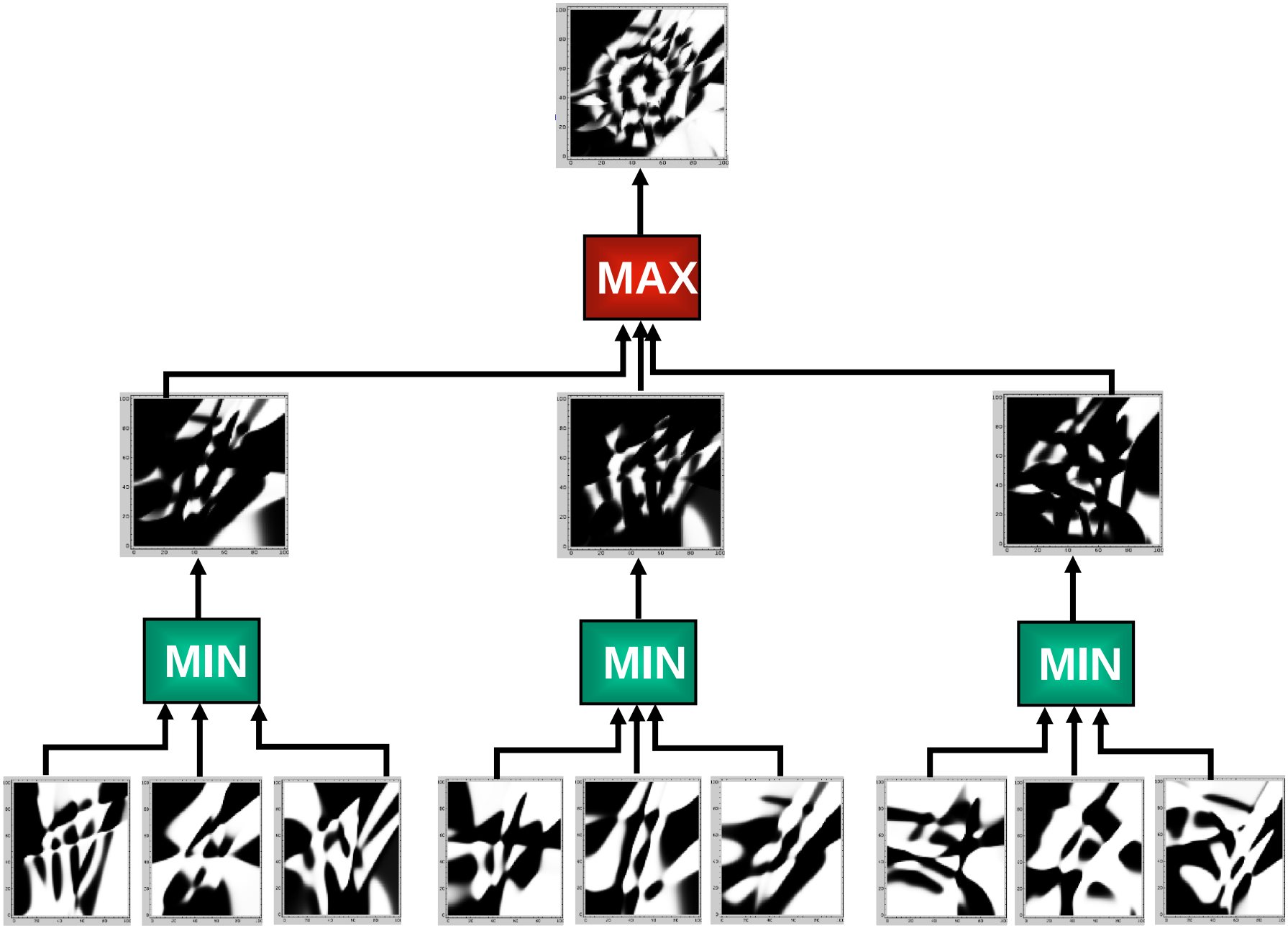
Decomposition of the Two-Spirals Problem

Two-Spirals Problem:

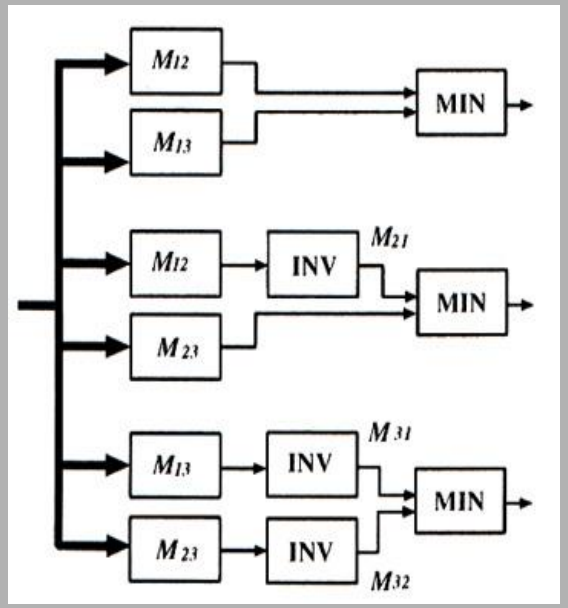


Massively Parallel Learning

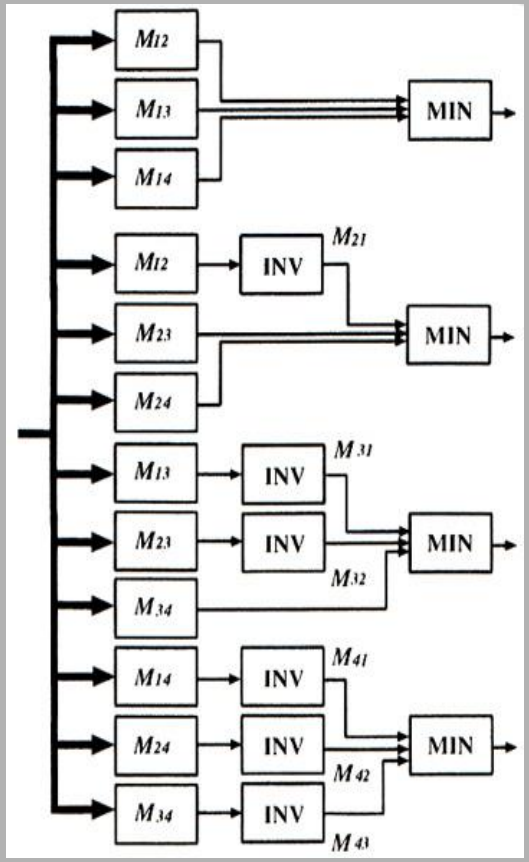



MAX
MIN
MIN
MIN

Min-Max Modular Neural Networks



Three-class



Four-class

Task Decomposition

- Training data for a K-class problem

$$T = \{ (X_l, Y_l) \}_{l=1}^{L}$$

- Decompose a K-class problem into K(K-1)/2 two-class problems

$$X_i = \{ X_l^{(i)} \}_{l=1}^{L_i} \quad \text{for } i = 1,2,\ldots,\mathrm{K}$$

$$T_{ij} = \{ (X_l^{(i)}, 1-\varepsilon) \}_{l=1}^{L_i} \cup \{ (X_l^{(j)}, \varepsilon) \}_{l=1}^{L_j} \text{ for } i = 1,\ldots K \text{ and } \; j = i+1$$

- Decompose a two-class problem into a number of relatively balanced two-class problems as smaller as needed

$$\text{Partition of } X_i \text{ into } N_i \text{ subsets } \quad X_{ij} = \{ X_l^{(ij)} \}_{l=1}^{L_i^{(j)}} \quad \text{for } \; j = 1,\ldots,N_i$$

$$T_{ij}^{(u,v)} = \{ (X_l^{(iu)}, 1-\varepsilon) \}_{l=1}^{L_i^{(u)}} \cup \{ (X_l^{(jv)}, \varepsilon) \}_{l=1}^{L_j^{(v)}}$$

$$\text{for } \; u = 1,\ldots,N_i, v = 1,\ldots,N_j, \text{ and } \; j \neq i$$

Number of Two-class Problems

- Number of smaller two-class problems

$$\sum_{i=1}^{K-1} \sum_{j=i+1}^{K} N_i \times N_j$$

N_i is the number of subsets for class C_i

- Number of training data for each of the two-class subproblems is about

$$\lceil L_i / N_i \rceil + \lceil L_j / N_j \rceil$$

L_i is the number of training data for class C_i

Time Complexity Analysis

- **Empirical observation(T. Joachims, 2002):**

$$O((l^+ + l^-)^c) \qquad c \text{ is domain-specific.}$$

- **Time complexity of M³-SVM in a parallel way:**

$$O\left(\left(\left\lfloor \frac{l^+}{N^+} \right\rfloor + \left\lfloor \frac{l^-}{N^-} \right\rfloor\right)^c\right)$$

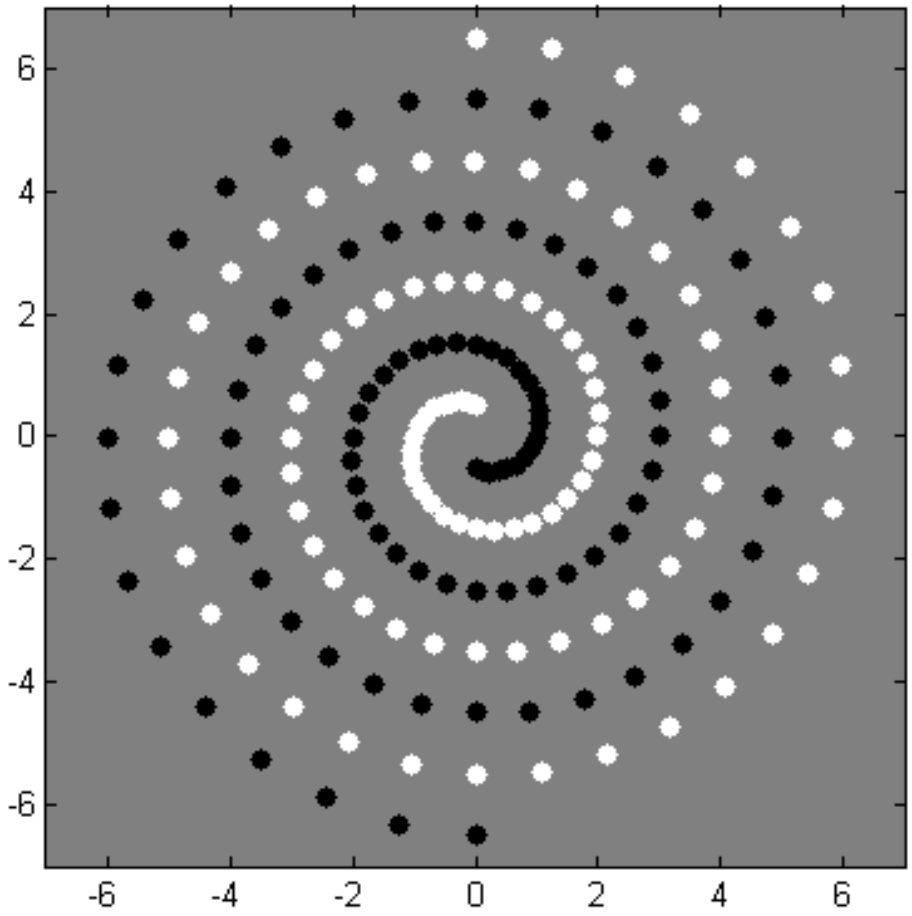
- **Time complexity of M³-SVM in a serial way:**

$$O\left(\frac{N^2}{N^c}\left(l^+ + l^-\right)^c\right) \qquad \text{suppose } N^+ = N^- = N$$

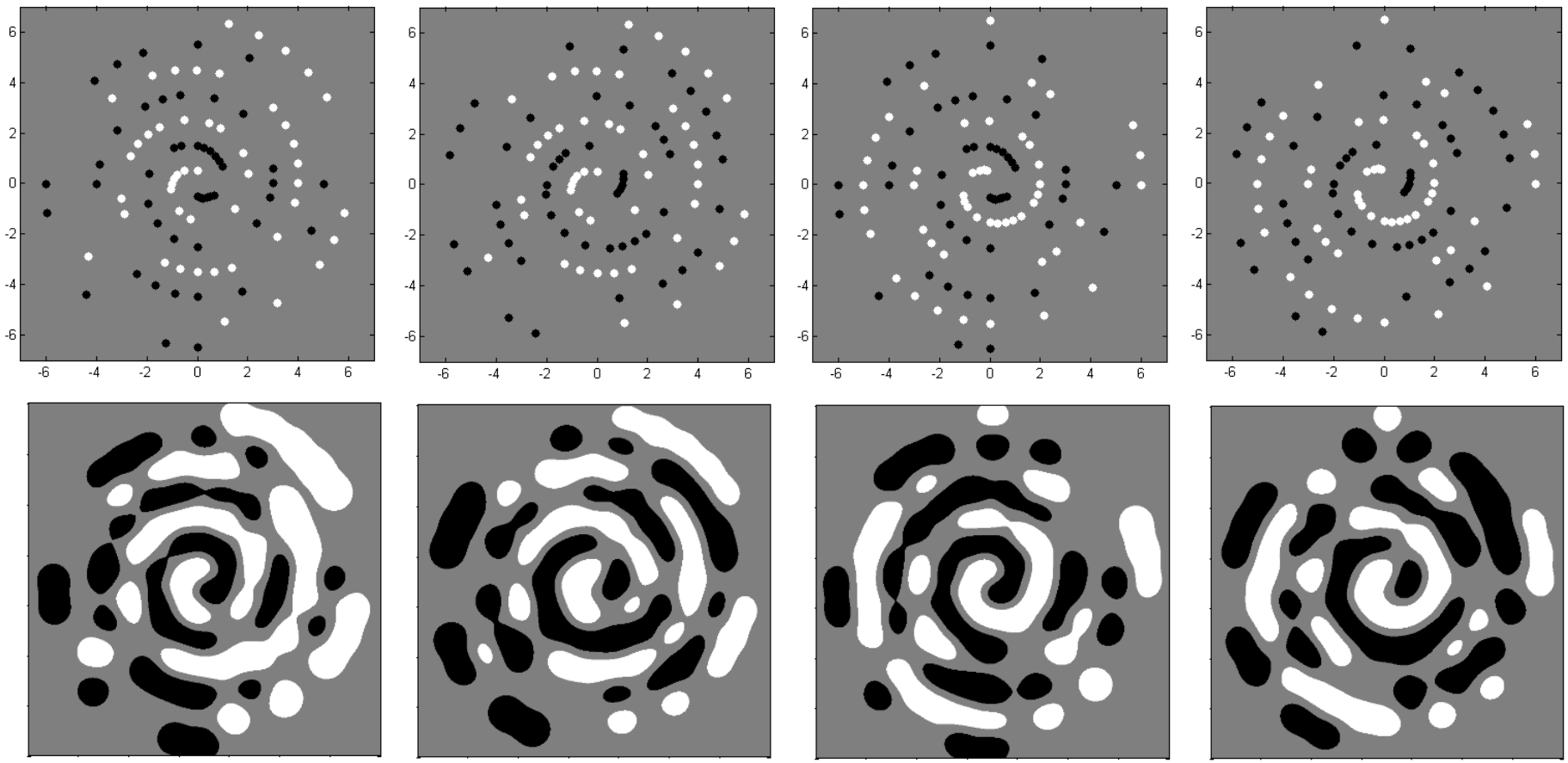
Task Decomposition Method

- How to divide training data set effectively is an important issue in M^3–SVM.
- Divide training data set randomly is a simple and straightforward approach.
- The geometric relation among the original data set may be damaged.

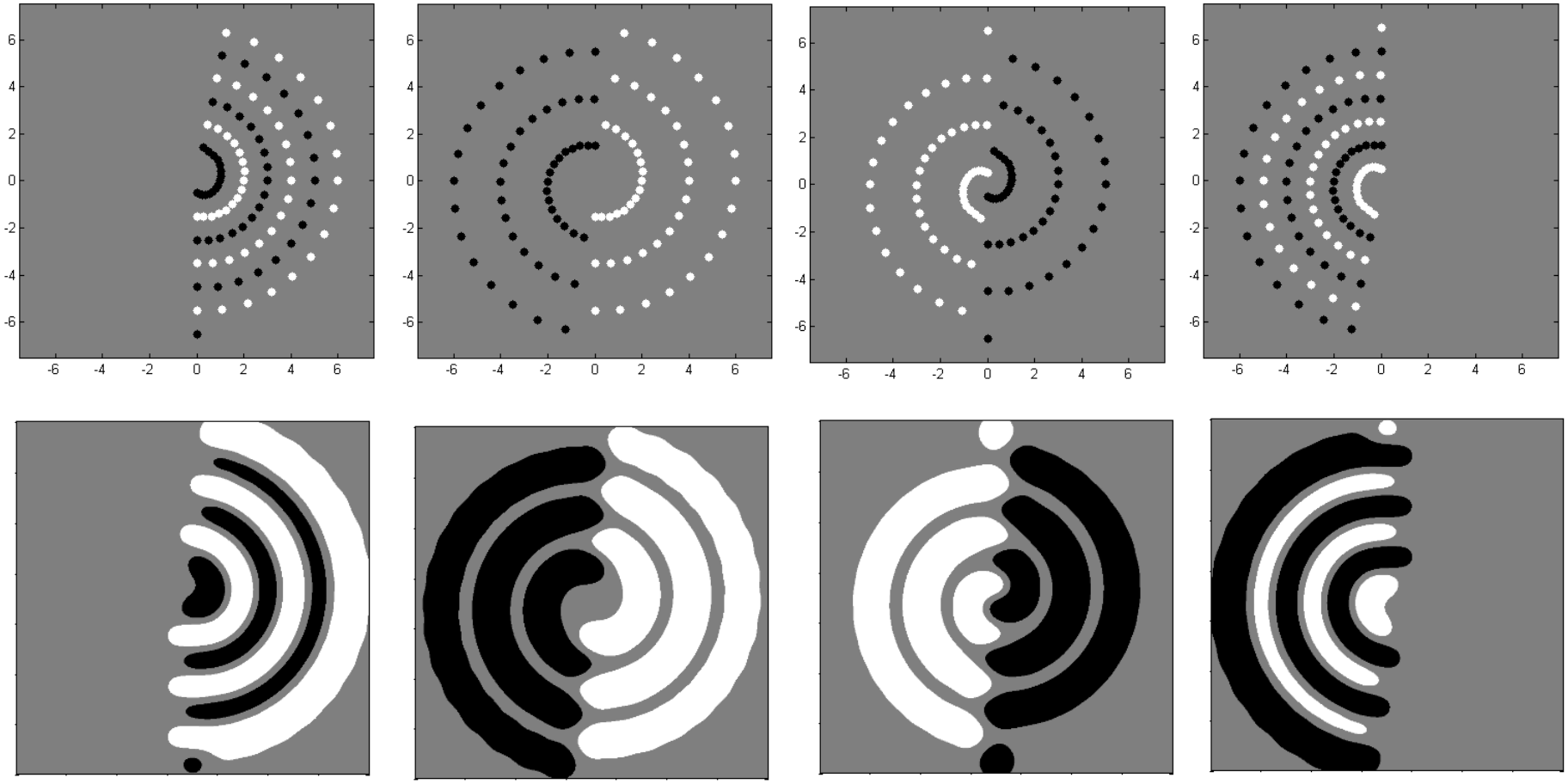
Two Spirals Problem



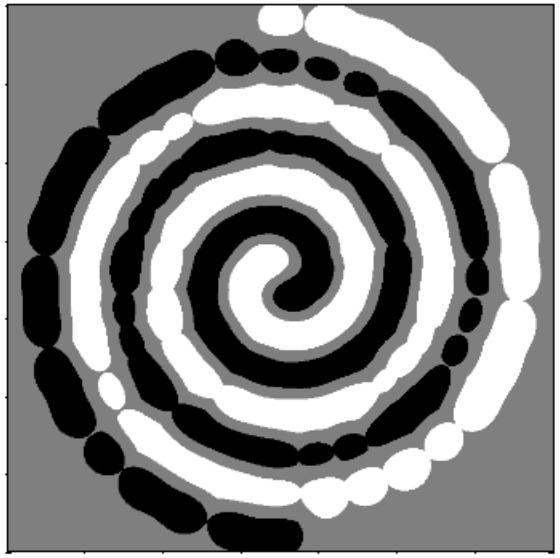
Random Partition



Y-axis partition



Boundaries of M3 Using Three Methods



random

Y-axis

Y-axis & overlap*

*overlap means two subsets share the training data along the y-axis

Hyperplane method

For more complicated high-dimensional problems, we divide the training data set using a hyperplane such as

$$\boxed{a_1 z_1 + a_2 z_2, \ldots, a_n z_n = b}$$

However, we never need to construct hyperplanes explicitly. Suppose we divide the training data set of class C_i to N_i subsets.

Hyperplane Method (cont.)

- Compute the distance between each training sample x of class Ci and basic hyperplane as follows,

$$H : a_1 z_1 + a_2 z_2, \ldots, a_n z_n = 0$$

$$\boxed{dist(x, H) = \frac{a_1 x_1 + a_2 x_2, \ldots, a_n x_n}{\sqrt{a_1^2 + a_2^2, \ldots, a_n^2}}}$$

Here, we take the direction of distance into account, the training data with positive value is located in the different side compared with the training data with negative value.

Hyperplane Method (cont.)

- Sort the training data according to the values of *dist*(x,H), namely, rearrange the training data according to their space position.
- Divide the reordered sequence of training data to N_i parts almost equally.

An Example of Hyperplane Method

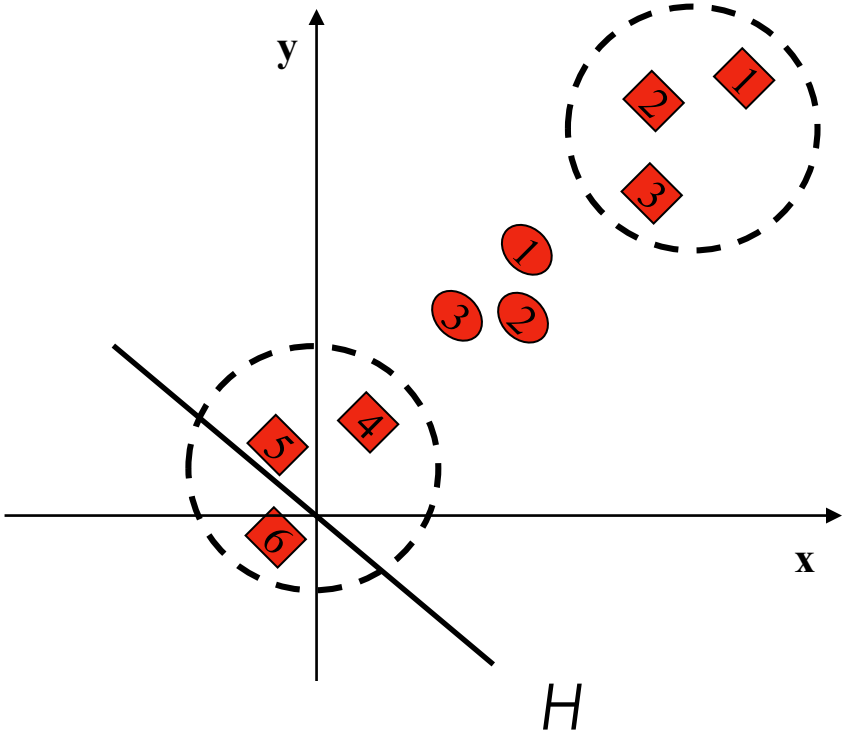
Here, we use hyperplane

$$H : z_1 + z_2, \ldots, z_n = 0$$

as basic hyperplane

We compute the distance and reordere the training data as the figure shows.

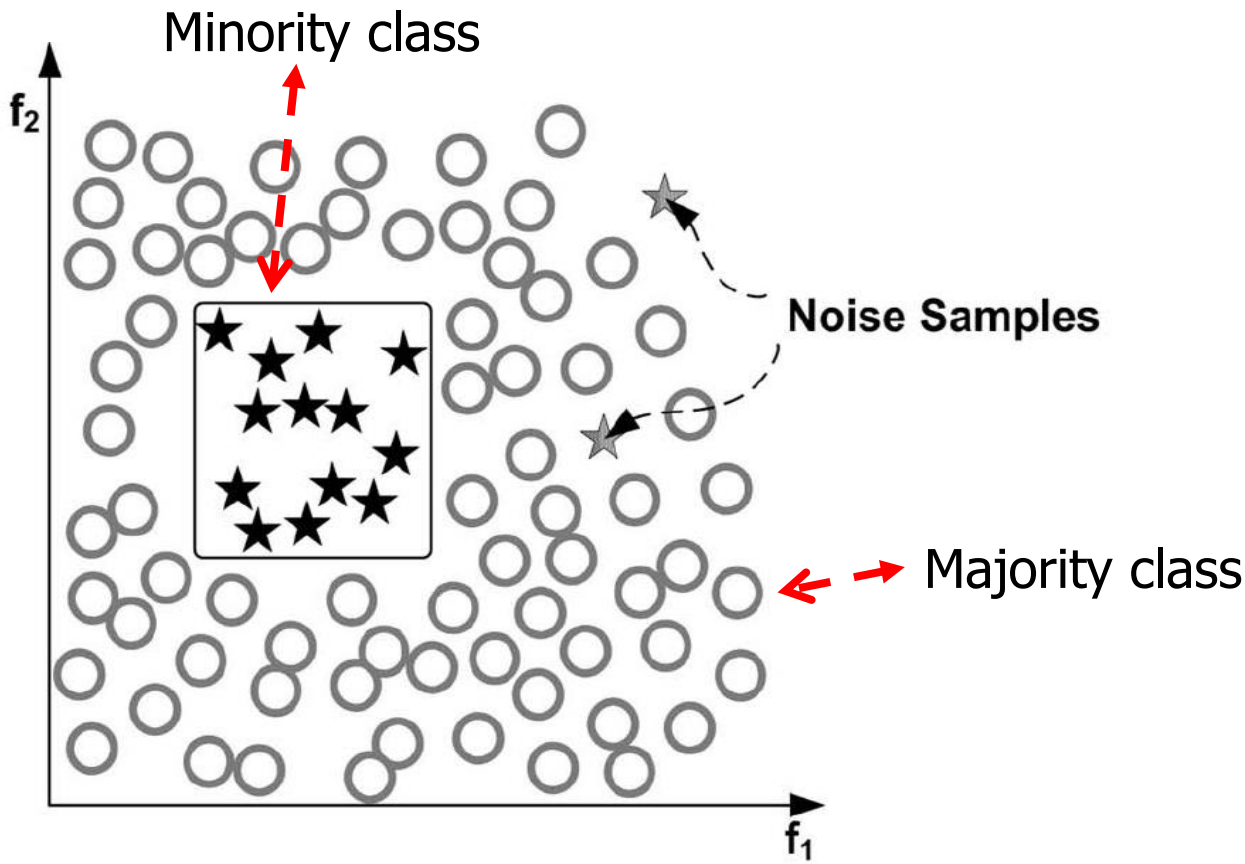
Then we divide the training data sets into two sets including 1,2,3 and 4,5,6 respectively.



Imbalanced data Set

- In a multi-class classification problem, any data set that exhibits an unequal distribution between its classes can be considered imbalanced
- In a two-class classification problem, a data set is said to present a class imbalance if it contains more training examples to one class (majority class) than the other (minority class).
- The data sets from real-world applications are usually imbalanced
- There are two kinds of imbalance forms:
  - Between-class imbalance
  - Within-class imbalance

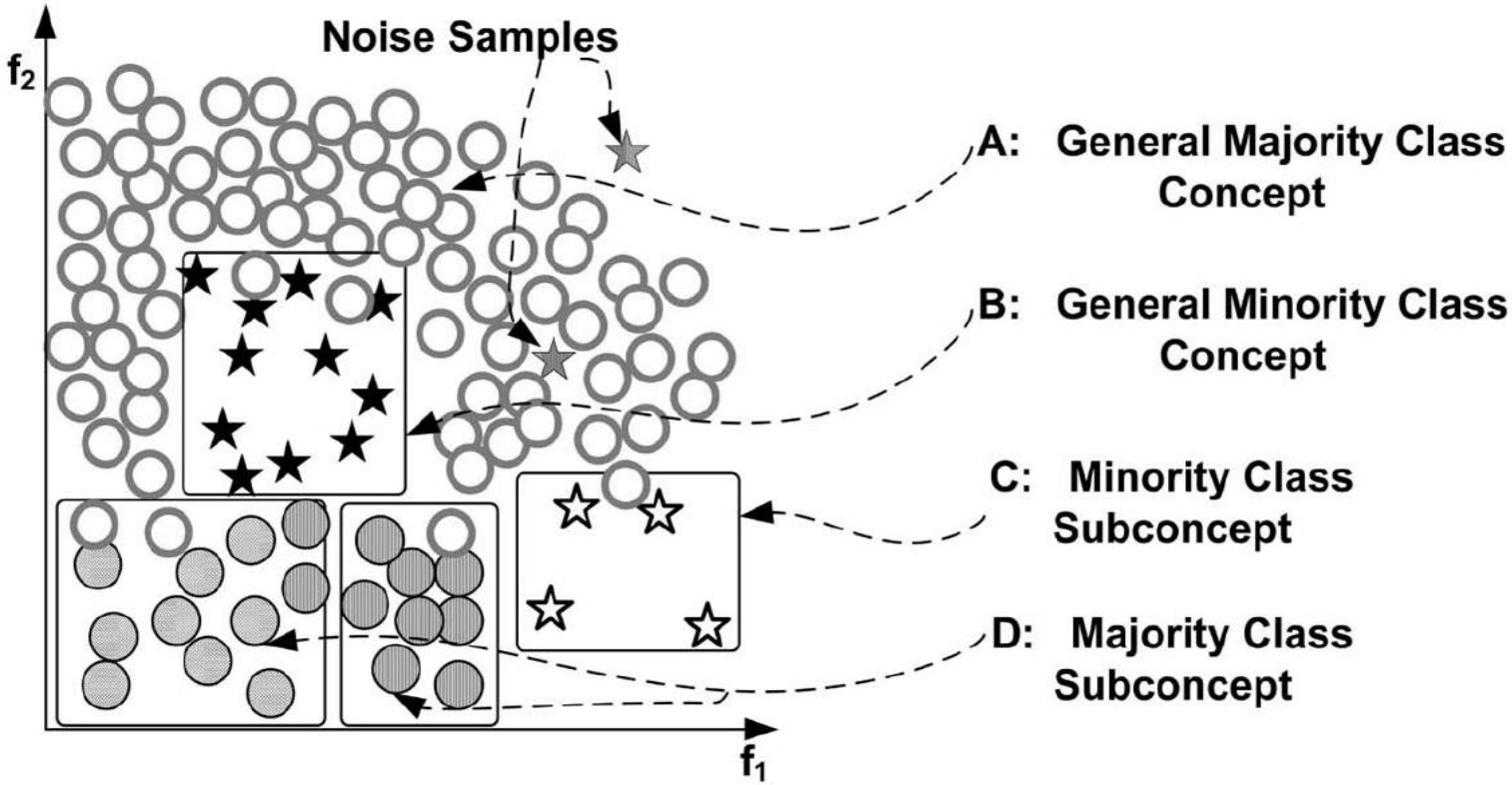
Between-class imbalance



The stars and circles represent the minority and majority classes, respectively

(He and Garcia, 2009)

Within-class imbalance



A high complexity data set with both between-class and within-class imbalances, multiple concepts, overlapping, noise, and lack of representative data.

(He and Garcia, 2009)

Reasons of leading to imbalances

- Rare instance: data for minority class are very limited, i.e., where the target concept is rare!
- Imbalances of this form are referred to as intrinsic
- Man-made imbalance: imbalanced data set is generated artificially
- Imbalances of this form are referred to as extrinsic
- Imbalances are caused by combination of intrinsic and extrinsic factors

Yeast Protein Sequences

Location	Sequence No.	Location	Sequence No.
Actin	29	Lipid particle	19
Bud	23	Microtubule	20
Bud neck	60	Mitochondrion	494
Cell periphery	106	Nuclear periphery	59
Cytoplasm	1576	Nucleolus	157
Early Golgi	51	Nucleus	1333
Endosome	43	Peroxisome	20
ER	272	Punctate composite	123
ER to Golgi	6	Spindle pole	58
Golgi	40	Vacuolar membrane	54
Late Golgi	37	vacuole	129

Reasons of leading to imbalances

- Rare instance: data for minority class are very limited, i.e., where the target concept is rare!
- Imbalances of this form are referred to as <u>intrinsic</u>
- Man-made imbalance: imbalanced data set is generated artificially
- Imbalances of this form are referred to as <u>extrinsic</u>
- Imbalances are caused by combination of intrinsic and extrinsic factors

Man-made imbalanced data sets

- One-versus-all (OVA)
  - K-class problem → K two-class sub-problems

$$X_i = \left\{ X_l^{(i)} \right\}_{l=1}^{L_i} \quad \text{for } i = 1, 2, \ldots, \mathrm{K}$$

$$T_i = \left\{ (X_l^{(i)}, 1-\varepsilon) \right\}_{l=1}^{L_i} \cup \left(\bigcup_{j=1, j\neq i}^{K} \left\{ (X_l^{(j)}, \varepsilon) \right\}_{l=1}^{L_j} \right) \text{ for } i = 1, \ldots K$$

- After this task decomposition, the original balanced problem becomes an imbalanced one and imbalanced problems becomes serious imbalanced ones
- For example, the order of between-class imbalance is 1:588 (6:3530) for Yeast Protein Sequences problem

Problems of imbalanced learning

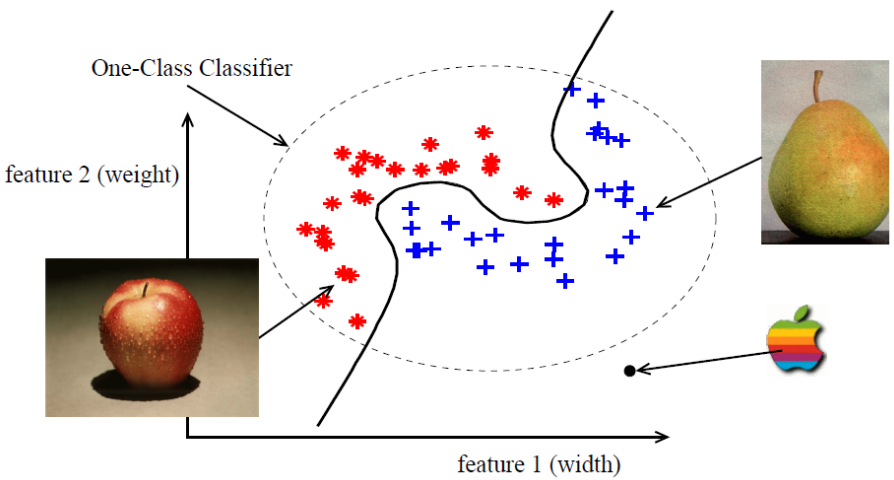
- The problem with class imbalance is that conventional classifiers are often biased towards the majority class
- Data from majority class are well-classified, whereas data from the minority class tend to be misclassified
- The reason is that standard classifiers attempt to reduce global quantities such as the error rate, not taking the data distribution into learning
- However, the most important knowledge usually resides in the rare cases

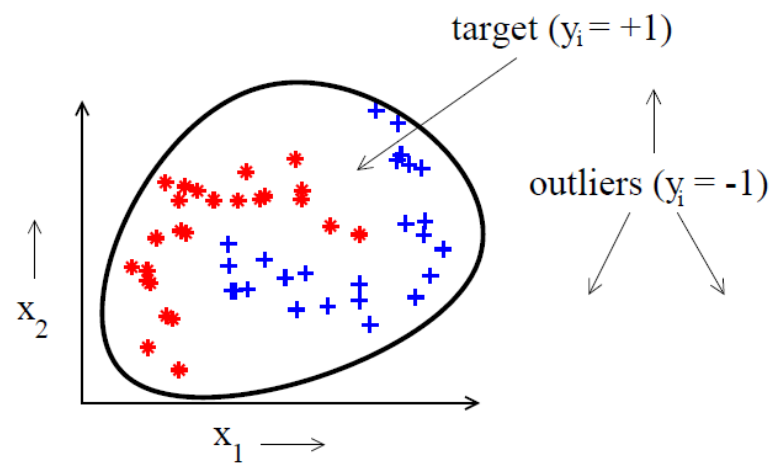
An example: Mammography Data Set

- A collection of images acquired from mammography exams
- Positive and negative samples represent cancerous patient and healthy patient, respectively
- The data set contains 10923 negative samples and 260 positive samples
- Classifiers have accuracy near to 100 percent on the majority class and accuracies of 0–10 percent on minority class
- Suppose a classifier achieves 10 percent accuracy on minority class
- As a result, 234 cancerous patients are classified into noncancerous patients

Existing Approaches

- Re-sampling Approach
  - Down-sampling
  - Over-sampling
- Cost-sensitive learning
- Active learning
- One-class classification

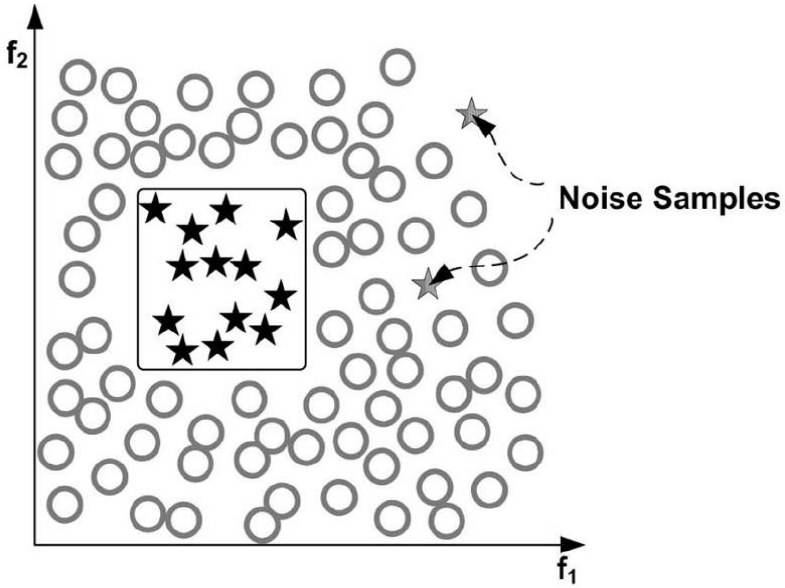




Re-sampling Approach

Convert an imbalanced data set into a balanced one by manipulating the samples

- Under-sampling : to remove majority samples from the original data set
- Over-sampling : to add minority samples to the original data set.



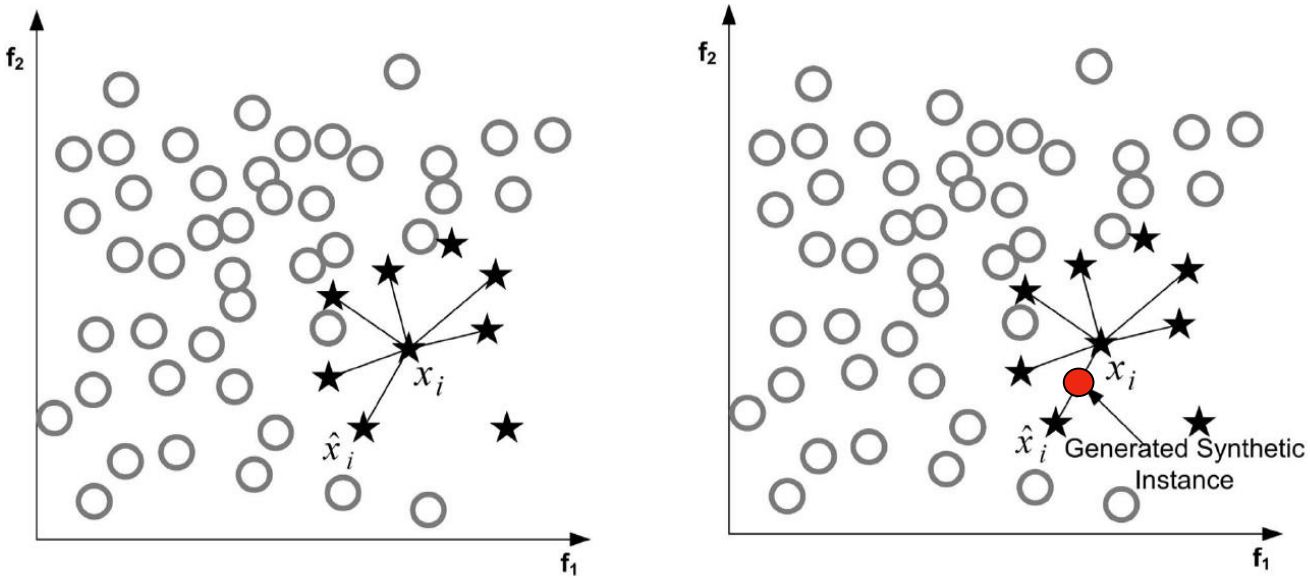
Re-sampling Approach: Down-Sampling

Make representative subsets of the majority class via data cleaning techniques

- the one-sided selection (OSS)
- the neighborhood cleaning rule (NCL)
- edited nearest neighbor (ENN)

Re-sampling Approach: Over-sampling

- Synthetic Minority Over-sampling Technique (SMOTE): create artificial minority samples



$$\boldsymbol{x_{new}} = \boldsymbol{x_i} + (\hat{\boldsymbol{x}}_i - \boldsymbol{x_i}) \times \delta$$

Cost-sensitive Learning

Associate high misclassifying cost to the minority classes

- Adacost: cost-sensitive weight updating strategy of Adaboost
- Cost-sensitive decision tree
- Cost-sensitive Bayesian classifiers

Active learning and one-class learning

- Active Learning:
  - Prefer the minority classes when interactively collecting new samples
- One-class Learning:
  - learn a class mainly through positive samples only
  - suitable for extremely imbalanced data sets

Remarks (I)

- Re-sampling Approach: manipulate the data
  - advantage: simplicity
  - disadvantage : under-sampling does not use the whole data set
- Cost-Sensitive Learning: adapt the classifiers
  - advantage: better performance
  - disadvantage : depend on the classifiers

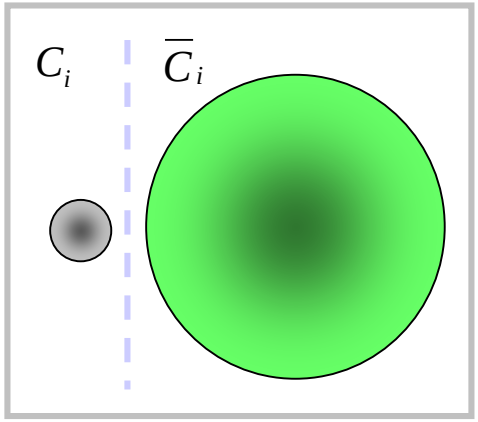
Remarks (II)

Active learning and one-class learning

- Almost unaffected by the sample ratio among classes
- Particularly suitable for extremely imbalanced problem where both re-sampling approaches and cost-sensitive learning fail

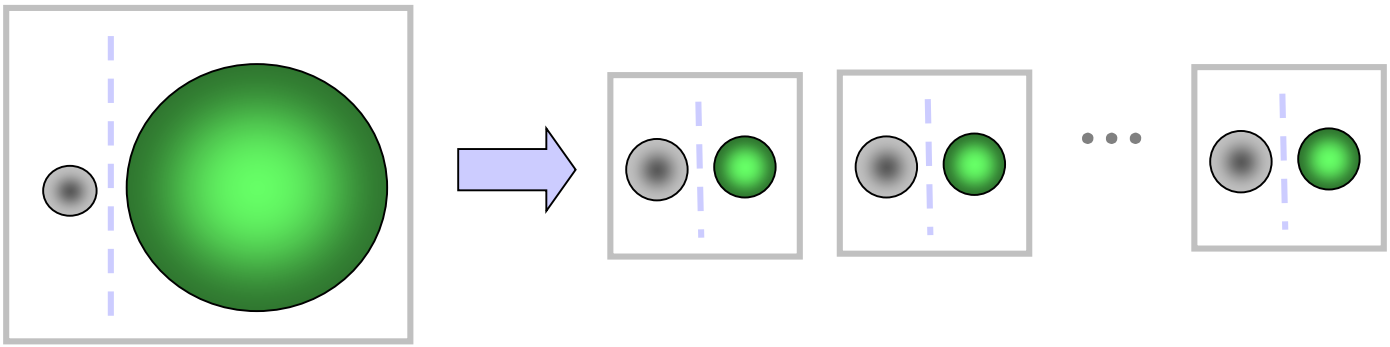
Main deficiencies of existing methods

- They are evaluated on small data sets
- Some of the two-class problems are very large and training classifiers on large-scale data sets are very time-consuming
- It is hard to incorporate prior knowledge into learning
- It is difficult to implement massively parallel learning

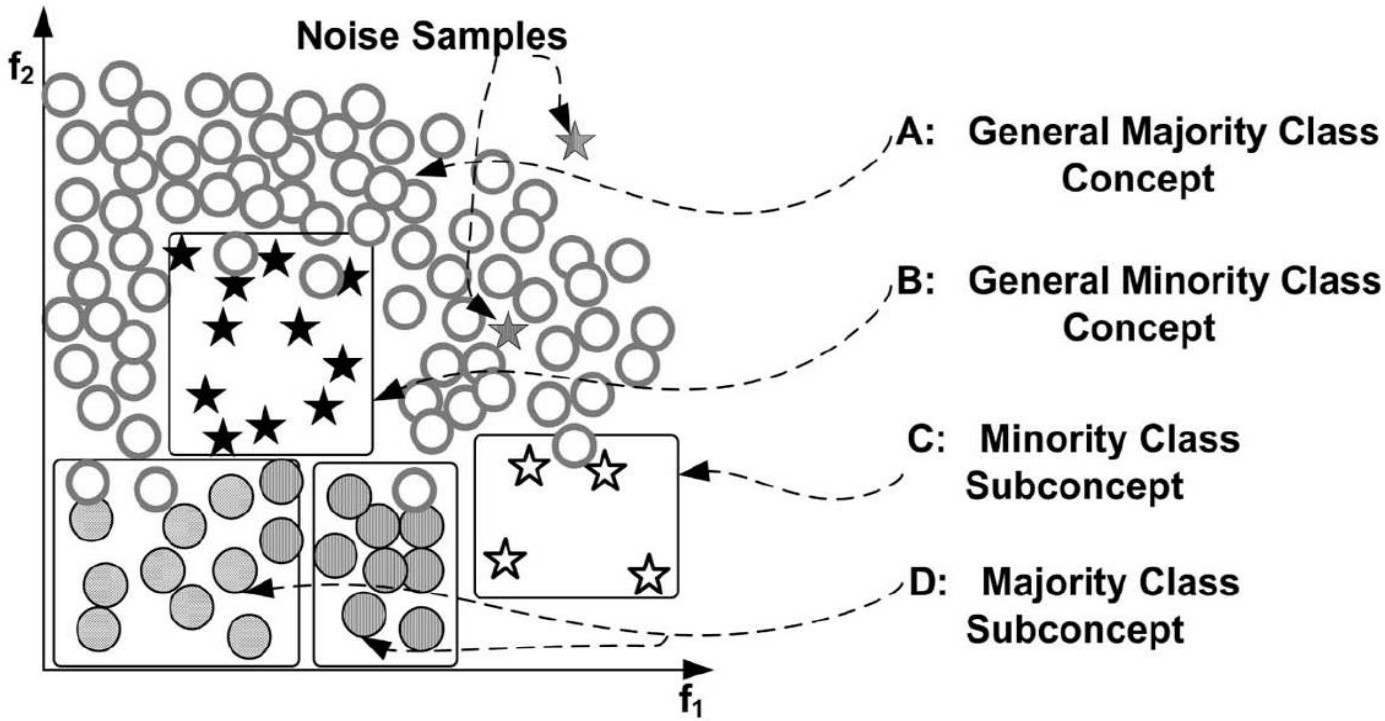


Basic idea of our approach

- Dispel imbalance by task decomposition instead of re-sampling
- Divide each of the imbalanced two-class problems into a number of relatively more balanced two-class subproblems with a part-versus-part task decomposition strategy
- Incorporate prior knowledge into learning by dividing training data



Why is task decomposition useful?

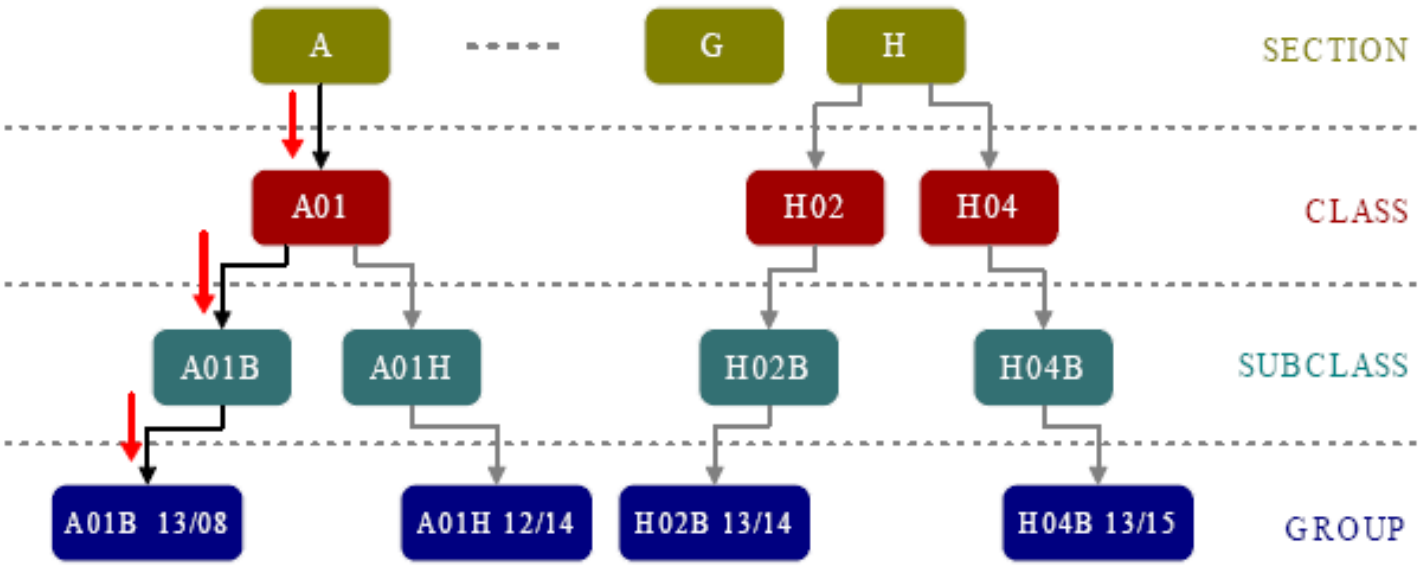


- A large-scale data set is organized following some rules
- These rules describes the data distribution in some degree
- Divide imbalanced data set using these rules as prior knowledge

An example of Japanese patent

	PATENT-JA-UPA-1998-000001
<Bibliography>	
[publication date]	(43)【公開日】平成１０年（１９９８）１月６日
[title of invenction]	(54)【発明の名称】土壌改良方法とその作業機
...	...
<Abstract>	
[purpose]	【課題】心土破砕、特に雪上心土破砕作業の際に積雪...
[solution]	【解決手段】心土破砕を行うために用いるサブソイウの...
...	...
<Claims>	
[claim1]	【請求項１】サブソイウ作業機を用いて心土破砕作業...
[claim2]	【請求項２】サブソイウ作業機において、そのナイフ...
...	...
<Description>	
[technique field]	【発明の属する技術分野】本発明は、土壌改良方法とそ...
[prior art]	【従来の技術】圃場の表面がまだ積雪に覆われている状...
[problem to be solved]	【発明が解決しようとする課題】心土破砕は通常春先に...
[means of solving problems]	【課題を解決するための手段】述のような目的達成す...
[effects of invention]	【発明の効果】以上の説明から明らかなように、本発明...
...	...
< Explanation of Drawing >	
[figure1]	【図１】本発明を施す圃場断面図である。
...	...

Structure vs distribution

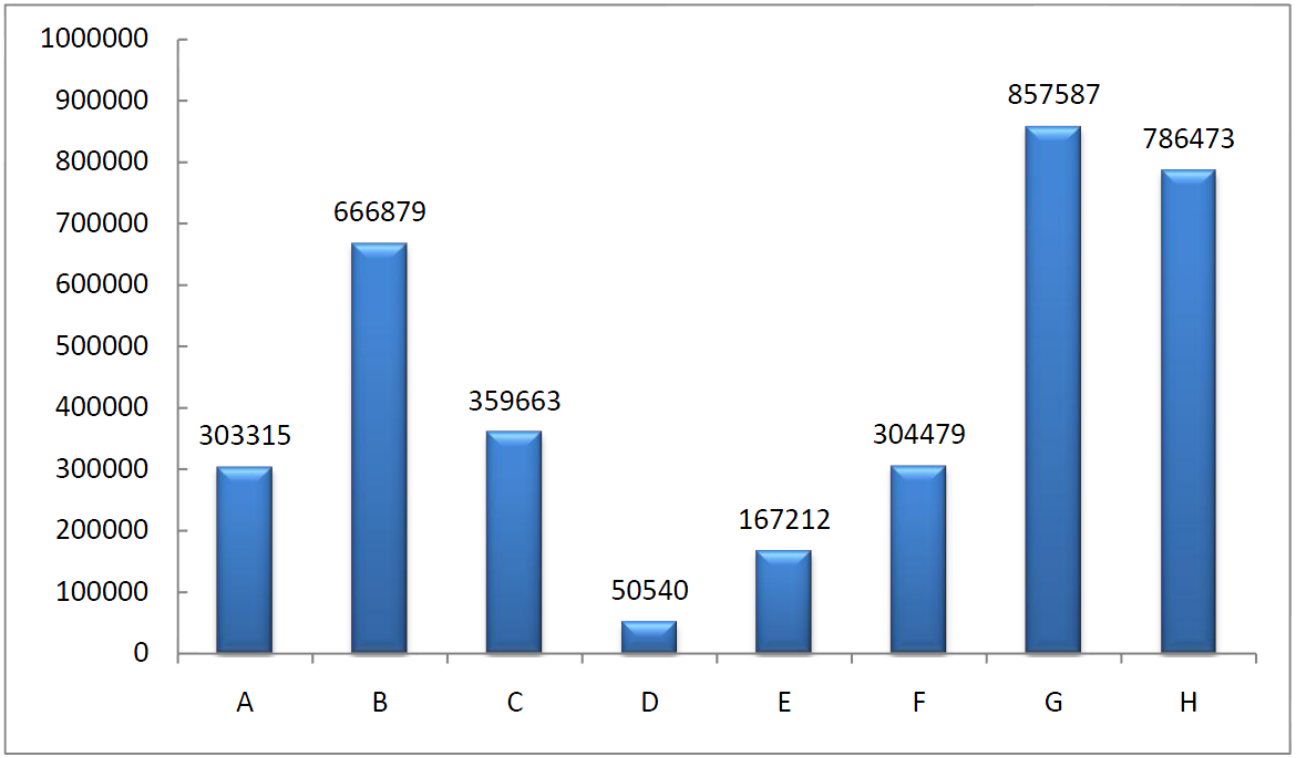


		Section	Class	Subclass	Group	Subgroup
No. Classes		8	120	630	7002	57913
No. Labels	Max	6	16	24	35	91
	Avg	1.3	1.5	1.7	2.2	2.7
No. Data	Max	857587	354104	176973	97008	23944
	Min	50540	38	1	1	1

Training and test data sets

# of years for training	years		set size	
	training	test	training	test
1	2000	2001,2002	358,072	733,570
2	1999,2000	2001,2002	714,004	733,570
3	1998–2000	2001,2002	1,055,391	733,570
4	1997–2000	2001,2002	1,386,850	733,570
5	1996–2000	2001,2002	1,727,356	733,570
6	1995–2000	2001,2002	2,064,325	733,570
7	1994–2000	2001,2002	2,415,236	733,570
8	1993–2000	2001,2002	2,762,563	733,570

Distribution of Japanese patents

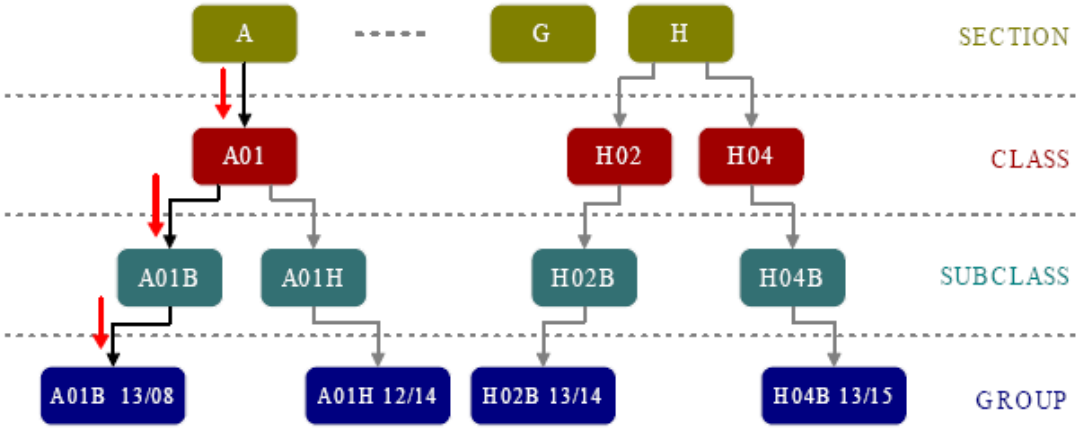


- The total number of Japanese patent applications from 1993 to 2002 is 3496137

Large-scale patent classification

Characteristics of patent classification

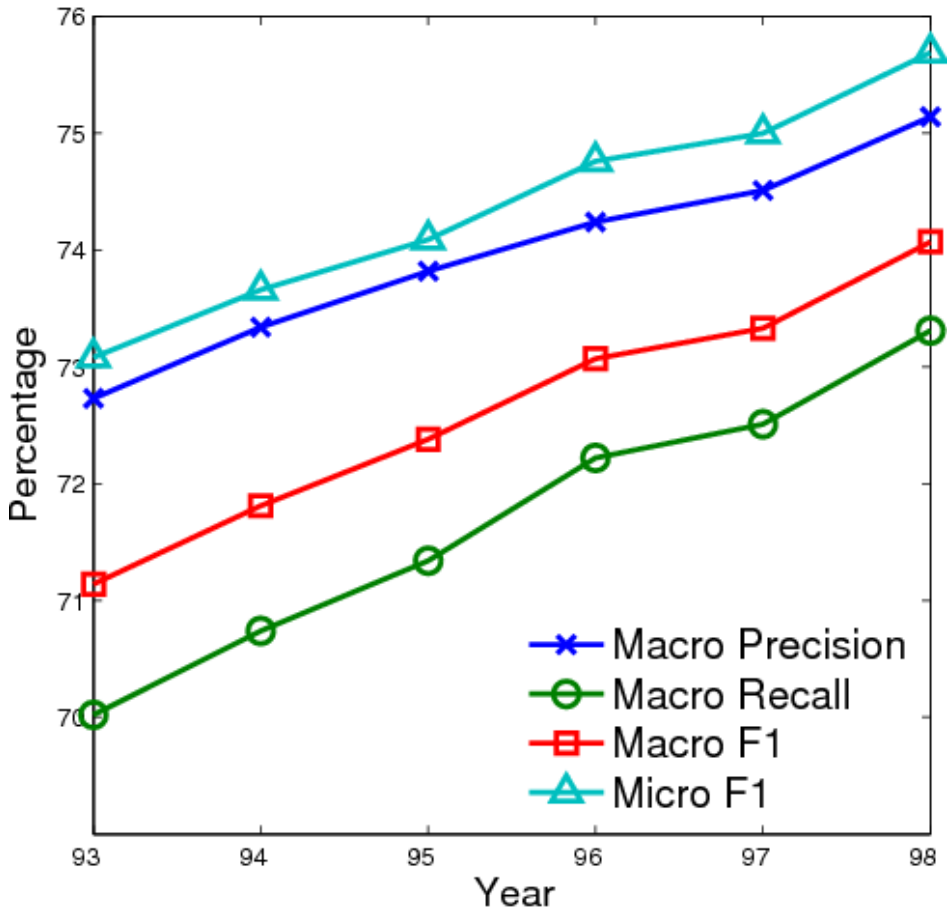
- Large-scale data set
- Imbalance
- Multi-label
- Hierarchical
- Time-varying feature



		Section	Class	Subclass	Group	Subgroup
No. Classes		8	120	630	7002	57913
No. Labels	Max	6	16	24	35	91
	Avg	1.3	1.5	1.7	2.2	2.7
No. Data	Max	857587	354104	176973	97008	23944
	Min	50540	38	1	1	1

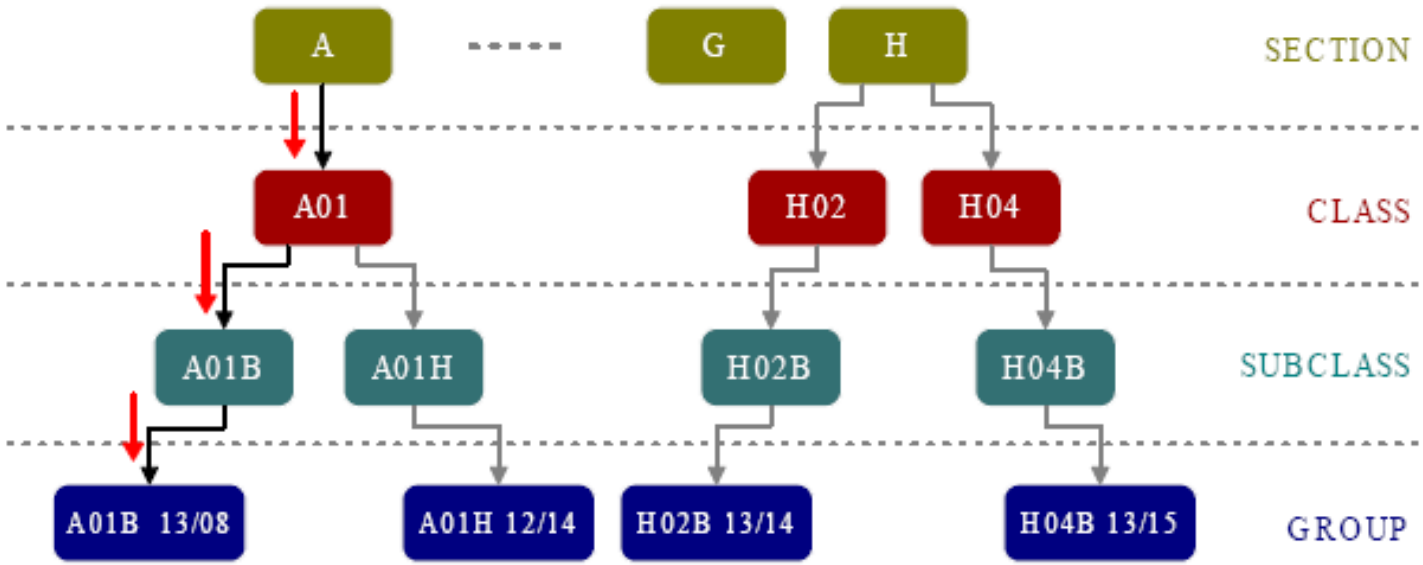
Our approach:
Incorporating both time and structure prior knowledge into training through task decomposition

Time-Varying Features of Patents



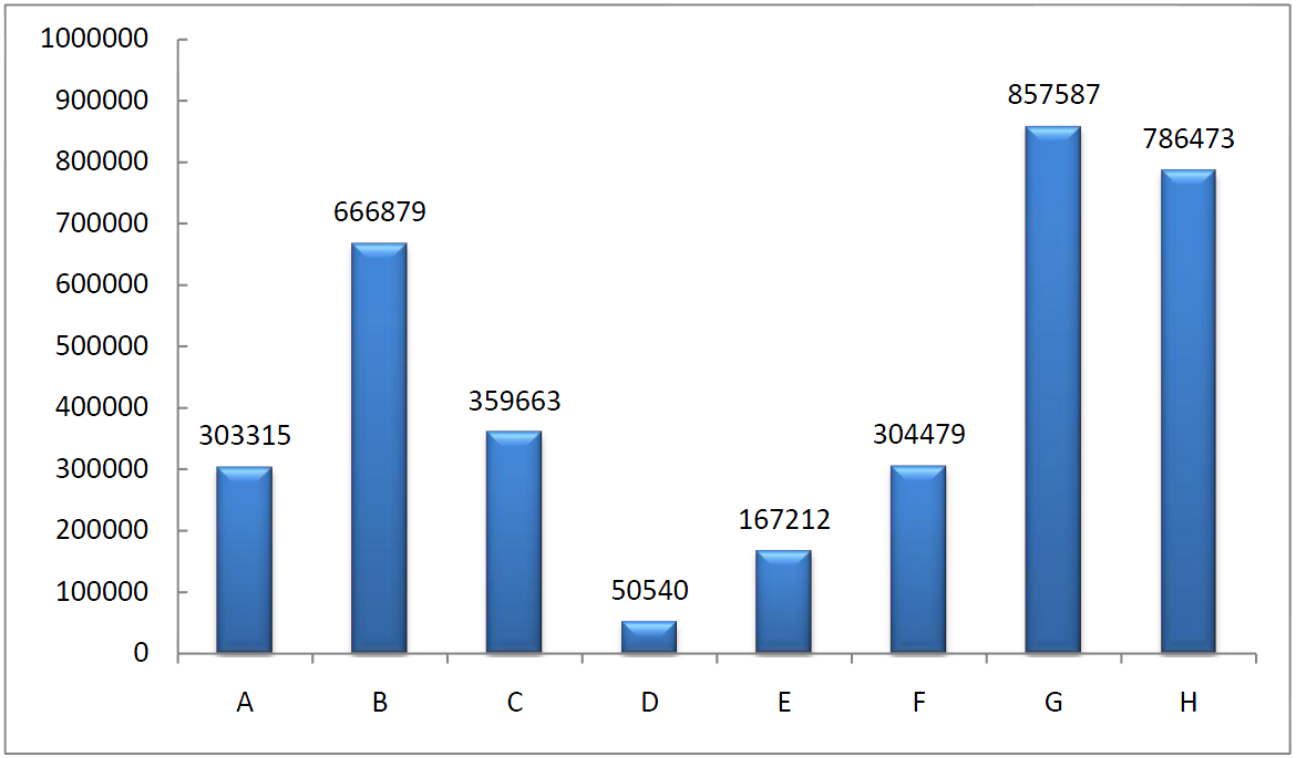
The data of 1999 are used as training data

Structure knowledge is useful!



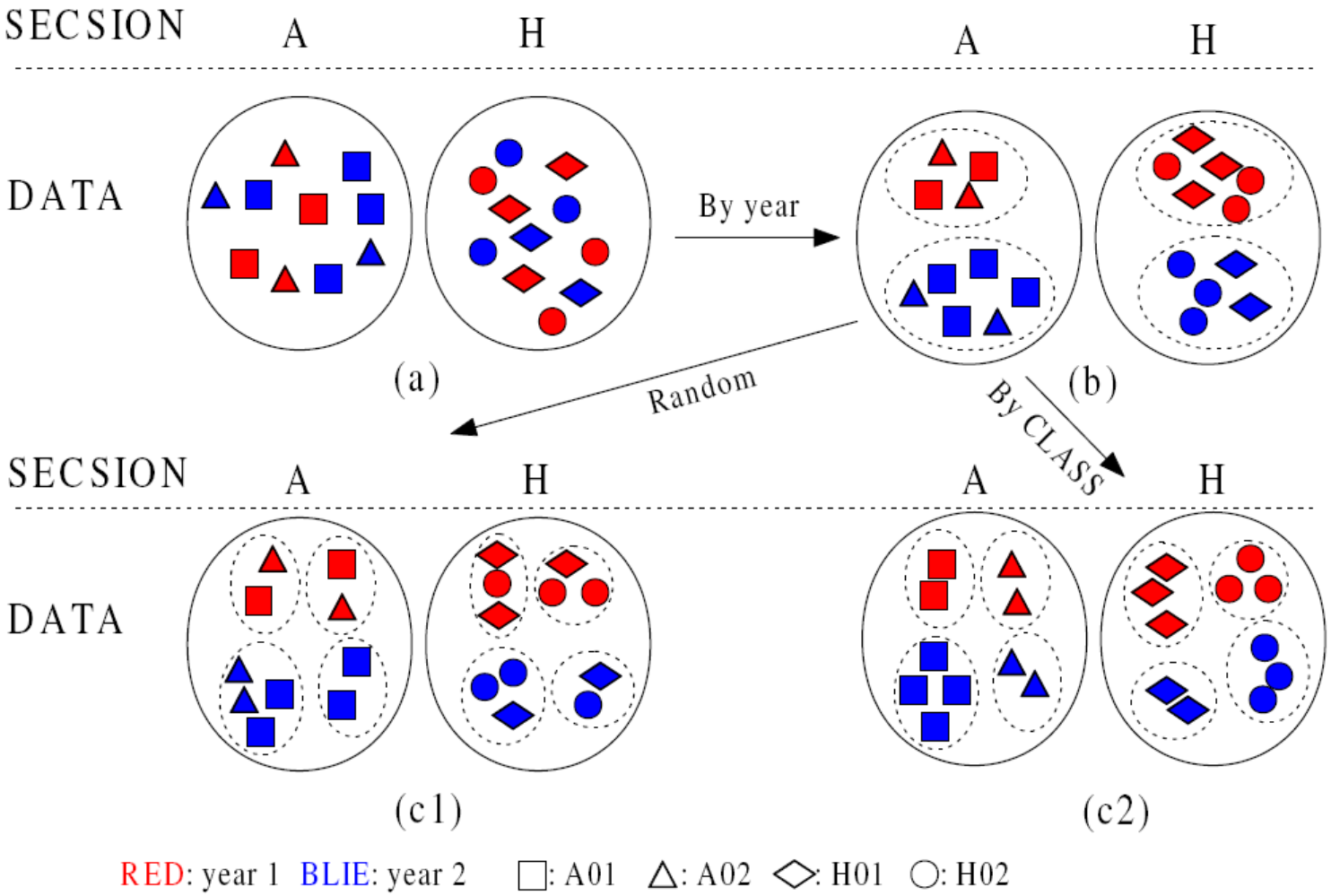
		Section	Class	Subclass	Group	Subgroup
No. Classes		8	120	630	7002	57913
No. Labels	Max	6	16	24	35	91
	Avg	1.3	1.5	1.7	2.2	2.7
No. Data	Max	857587	354104	176973	97008	23944
	Min	50540	38	1	1	1

Distribution of Japanese patents



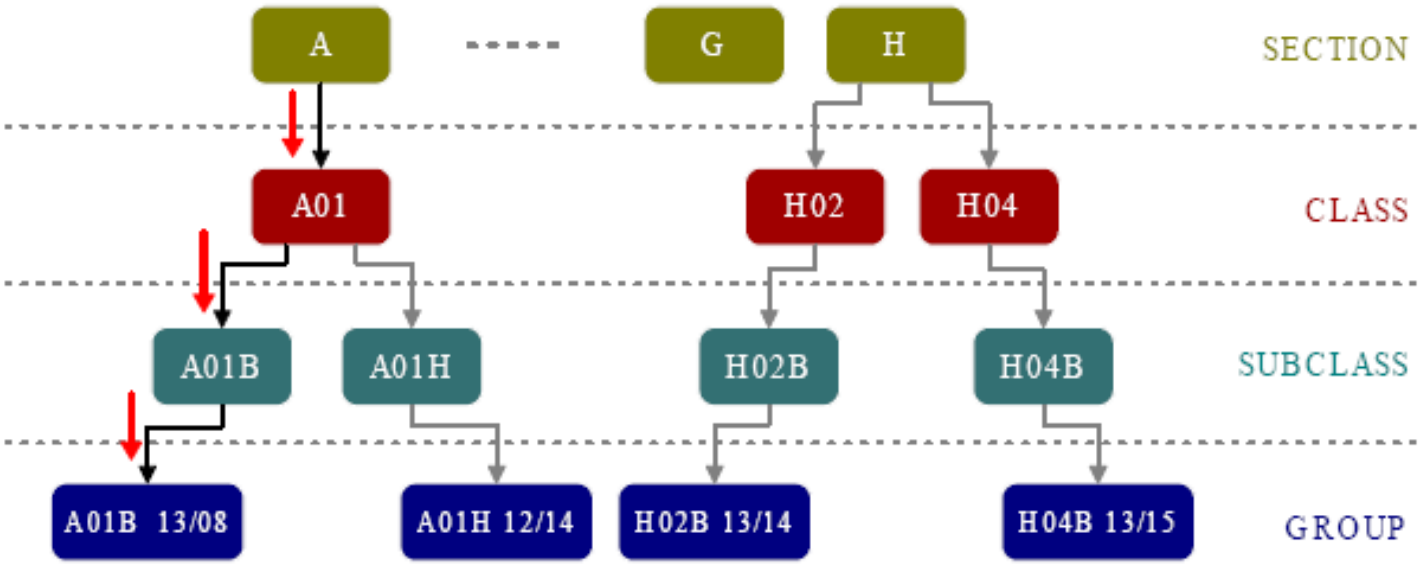
- The total number of Japanese patent applications from 1993 to 2002 is 3496137

Year-class decomposition strategy



Performance comparison

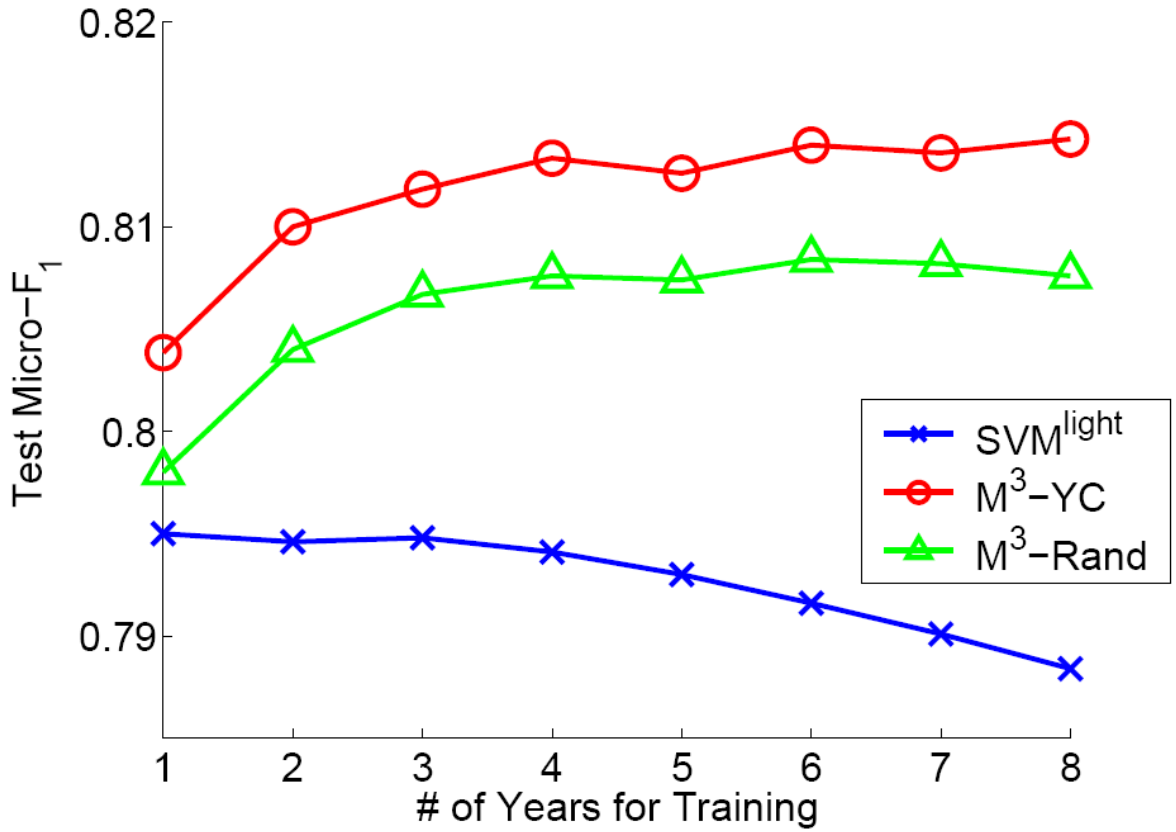
- R-M3-SVM: decompose data set randomly
- YC-M3-SVM: decompose data set by year and class
- Conventional SVMs are selected as a baseline



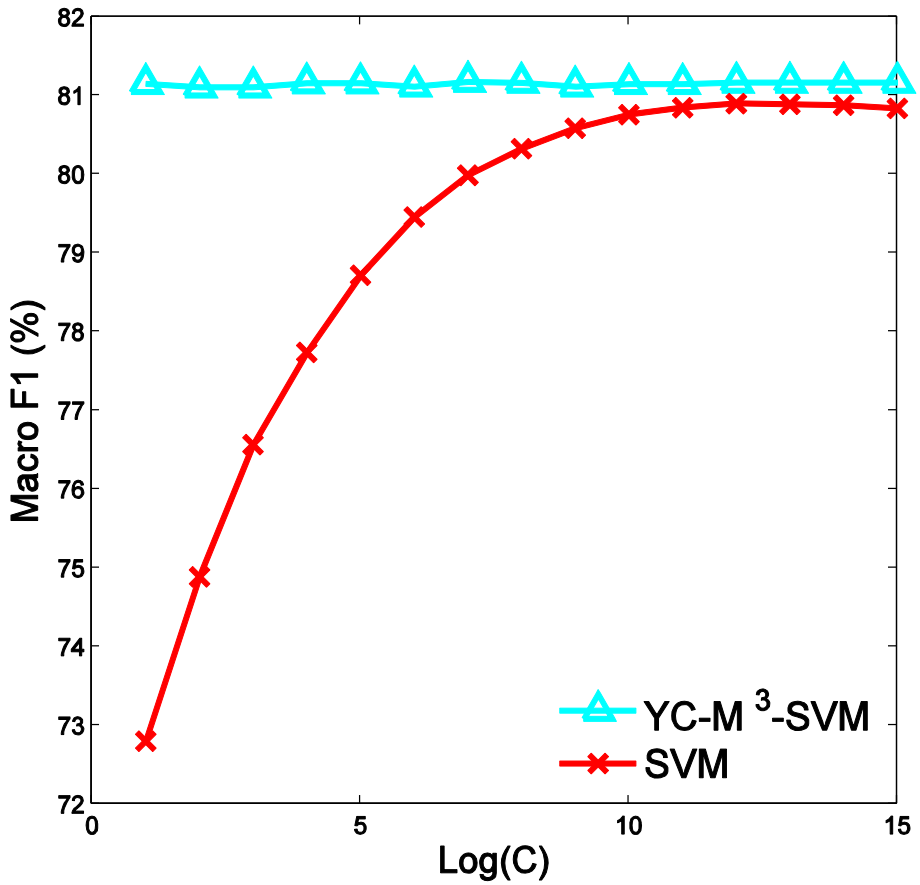
Training and test data sets

# of years for training	years		set size	
	training	test	training	test
1	2000	2001,2002	358,072	733,570
2	1999,2000	2001,2002	714,004	733,570
3	1998–2000	2001,2002	1,055,391	733,570
4	1997–2000	2001,2002	1,386,850	733,570
5	1996–2000	2001,2002	1,727,356	733,570
6	1995–2000	2001,2002	2,064,325	733,570
7	1994–2000	2001,2002	2,415,236	733,570
8	1993–2000	2001,2002	2,762,563	733,570

Comparison of classification accuracy



Variation with changes of C



Min-Max Modular Network with Gaussian-Zero-Cross (GZC) Function

(B. L. Lu, M. Ichikawa, 2002)

Task Decomposition

- Training data for a K-class problem

$$T = \{ (X_l, Y_l) \}_{l=1}^{L}$$

- Decompose a K-class problem into K(K-1)/2 two-class problems

$$X_i = \left\{ X_l^{(i)} \right\}_{l=1}^{L_i} \quad \text{for } i = 1,2,\ldots,\mathrm{K}$$

$$T_{ij} = \left\{ (X_l^{(i)}, 1-\varepsilon) \right\}_{l=1}^{L_i} \cup \left\{ (X_l^{(j)}, \varepsilon) \right\}_{l=1}^{L_j} \text{ for } i = 1,\ldots K \text{ and } \; j = i+1$$

- Decompose a two-class problem into a number of relatively balanced two-class problems as smaller as needed

$$\text{Partition of } X_i \text{ into } N_i \text{ subsets } \quad X_{ij} = \left\{ X_l^{(ij)} \right\}_{l=1}^{L_i^{(j)}} \quad \text{for } \; j = 1,\ldots,N_i$$

$$T_{ij}^{(u,v)} = \left\{ (X_l^{(iu)}, 1-\varepsilon) \right\}_{l=1}^{L_i^{(u)}} \cup \left\{ (X_l^{(jv)}, \varepsilon) \right\}_{l=1}^{L_j^{(v)}}$$

$$\text{for } \; u = 1,\ldots,N_i, v = 1,\ldots,N_j, \text{ and } \; j \neq i$$

Number of Two-class Problems

- Number of smaller two-class problems

$$\sum_{i=1}^{K-1} \sum_{j=i+1}^{K} N_i \times N_j$$

N_i is the number of subsets for class C_i

- Number of training data for each of the two-class subproblems is about

$$\lceil L_i / N_i \rceil + \lceil L_j / N_j \rceil$$

L_i is the number of training data for class C_i

Number of Linear Separable Problems

Total number of linear separable problems

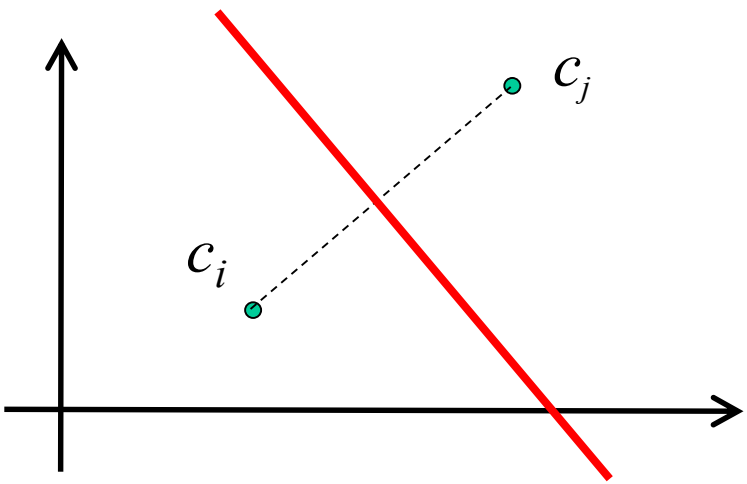
$$\boxed{\sum_{i=1}^{K} \sum_{j=1, j \neq i}^{K} L_i \times L_j}$$

Hyperplane

A useful hyperplane is the perpendicular bisector of the line joining the two training inputs c_i and c_j

$$f_{ij}(x) = (c_j - c_i)^t x + \frac{1}{2}(\|c_i\|^2 - \|c_j\|^2) = 0$$

where $||z||^2$ is the squared magnitude of the vector z.



Gaussian Zero-Crossing Discriminate Function

To overcome the weakness of linear discriminant function, a Gaussian zero-crossing discriminate function is proposed:

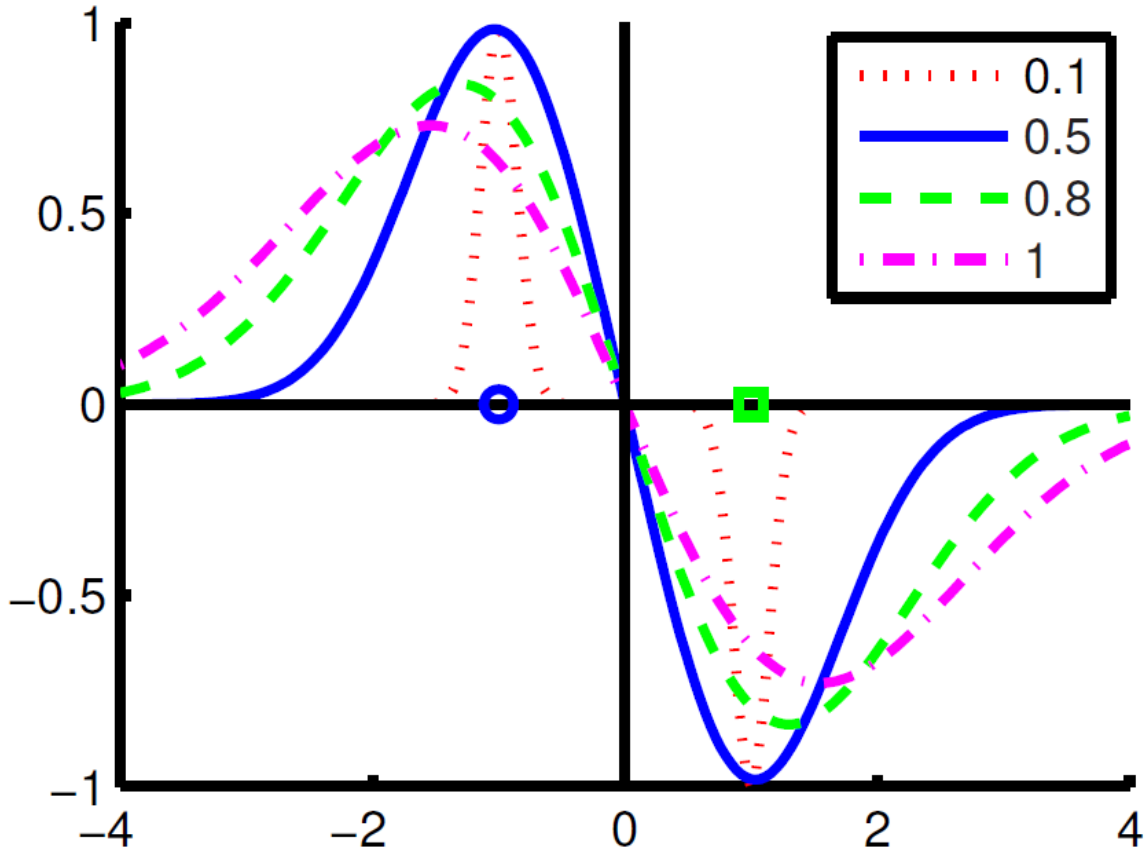
$$f_{ij}(x) = exp\left[-\left(\frac{\|x - c_i\|}{\sigma}\right)^2\right] - exp\left[-\left(\frac{\|x - c_j\|}{\sigma}\right)^2\right]$$

where x is the input vector, c_i and c_j are the given training inputs belonging to class C_i and class C_j ( ). $\sigma = \lambda || c_i - c_j ||$ and λ is a user-defined constant, which decides the shape of GZC functions

$i \neq j$

Gaussian zero-crossing discriminate function (2)

Shape of GZC functions with $\lambda = 0.1$, 0.5,0.8 and 1, respectively.



Output of M^3-GZC network

The output of M^3–GZC network is deifned as follows.

$$g_i(x) = \begin{cases} 1 & \text{if } y_i(x) > \theta_i \\ Unknown & \text{if } -\theta_j \leq y_i(x) \leq \theta_i \\ -1 & \text{if } y_i(x) < -\theta_j \end{cases}$$

where θ_i and θ_j are the threshold limits of class C_i and C_j, respectively, and y_i denotes the transfer function of M^3-GZC network for class C_i, which discriminates the patterns of class C_i from those of the rest of the classes.

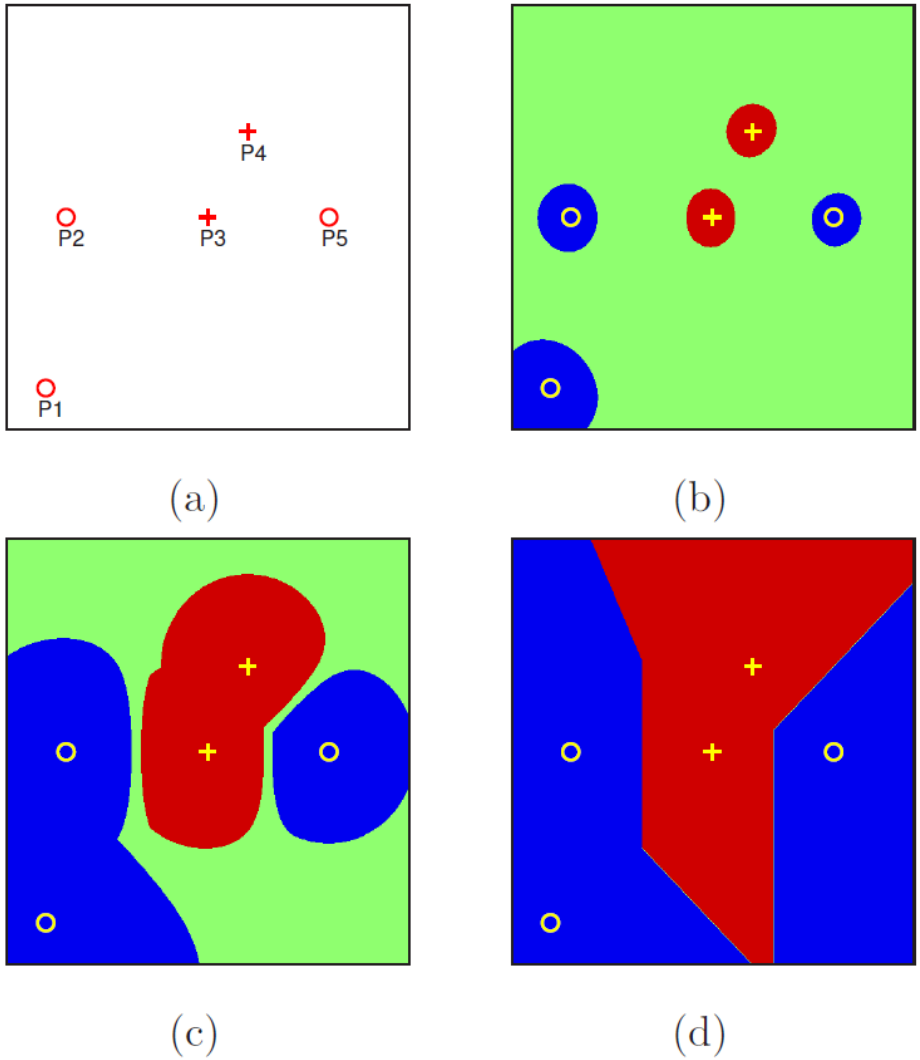


Figure 1.3. A two-class problem and the decision boundaries at different threshold limits, where P_i denotes the ith instance presented to the network and the **green area** denotes the '*Unknown*' region. (a) The training inputs of a two-class problem; (b), (c), (d) Decision boundaries at different threshold limits; (b) $\theta_i = \theta_j = 0.8$; (c) $\theta_i = \theta_j = 0.1$; and (d) $\theta_i = \theta_j = 0$.

Properties of M³-GZC Network
Highly Modular Structure

The total number of modules in a M³–GZC network is:

$$\sum_{i=1}^{K} \sum_{j=1, j \neq i}^{K} L_i \times L_j$$

Guarantee of Learning Convergence

- Many traditional neural network models use gradient-based methods on a differentiable error measure to find the solution, and therefore learning convergence can not be guaranteed and they are easy to be trapped in local minimum.
- Unlike these methods, the learning procedure of M3-GZC network is simply combining the solution of a reasonable number of linearly separable subproblems, so its learning convergence can be guaranteed in polynomial time.

Locally Tuned Response Characteristic

- The main difference between M3-GZC network and M3 network with linear discriminant function (M3-Linear) is that the former has the ability of saying `Unknown' to some inputs while the latter pretends to be omniscient to all the inputs.
- In a M3-GZC network, the interpolation and extrapolation capabilities can be easily controlled by selecting different values of threshold limits。
- The threshold limit can be viewed as a degree of confidence of correct classification.

Future Work

- Canonical neural computation:
  - A variety of anatomical, physiological, and behavioral evidence suggests that the brain performs computations using modules that are repeated across species, brain areas, and modalities. What are these canonical modules, and how can we elucidate their underlying circuitry and mechanisms?
- My hypothesis:

  both **minimization** and **maximization** are

  two canonical computation rules in our brain

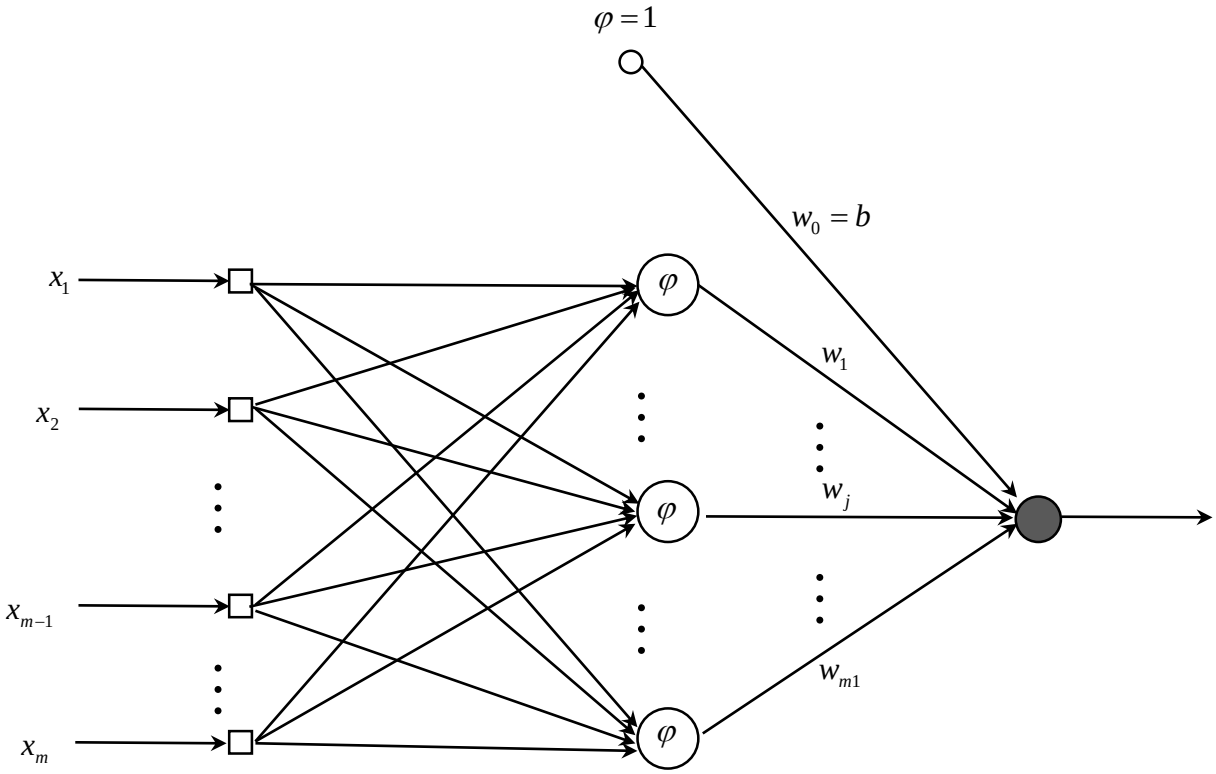
Radial-Basis Function Networks

Introduction to RBF Network

- A basic *radial-basis function (RBF) network* consists of three layers having entirely different roles:
  1. Input layer is made up of source nodes (sensory units).
  2. The hidden layer applies a nonlinear transformation from the input space to the hidden space.

     - RBF networks have only one, often high-dimensional hidden layer.
  3. A linear output layer.
- The hidden space is usually chosen high-dimensional because of two reasons:
  1. Pattern vectors are more likely to be linearly separable in a high-dimensional space.
  2. The approximation ability of the network is the better the more there are hidden units.

Radial-Basis Function (RBF) Network



Input layer

Hidden layer of m_1 radial-basis functions

Output layer

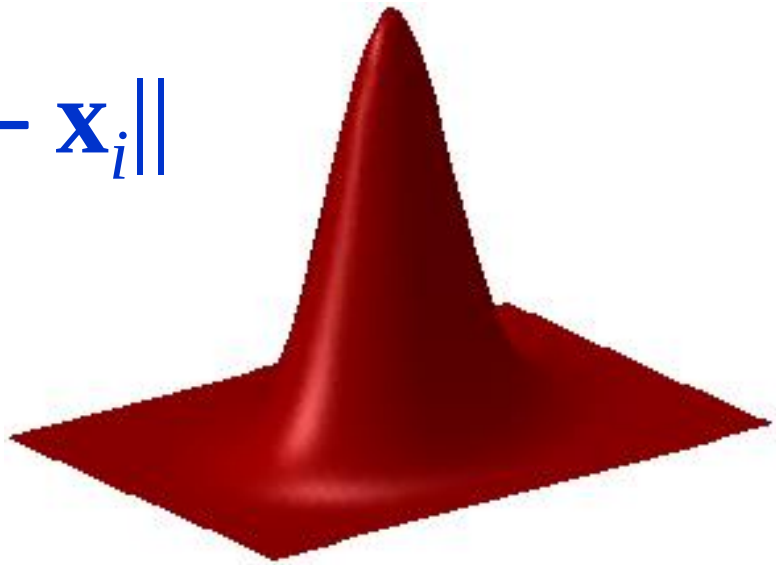
Radial Basis Function

$$f(\mathbf{x}) = \sum_{i=1}^{m} w_i \phi_i(\mathbf{x})$$

Three parameters for a radial basis function:

$$\phi_i(\mathbf{x}) = \phi\left(\|\mathbf{x} - \mathbf{x}_i\|\right)$$

- Center: $\mathbf{x}_i$
- Distance Measure: $r = \|\mathbf{x} - \mathbf{x}_i\|$
- Shape: ϕ



Typical Radial Functions

- **Gaussian:**

$$\phi(r) = e^{-\frac{r^2}{2\sigma^2}} \quad \sigma > 0 \quad \text{and} \quad r \in \Re$$

- **Multiquadratics:**

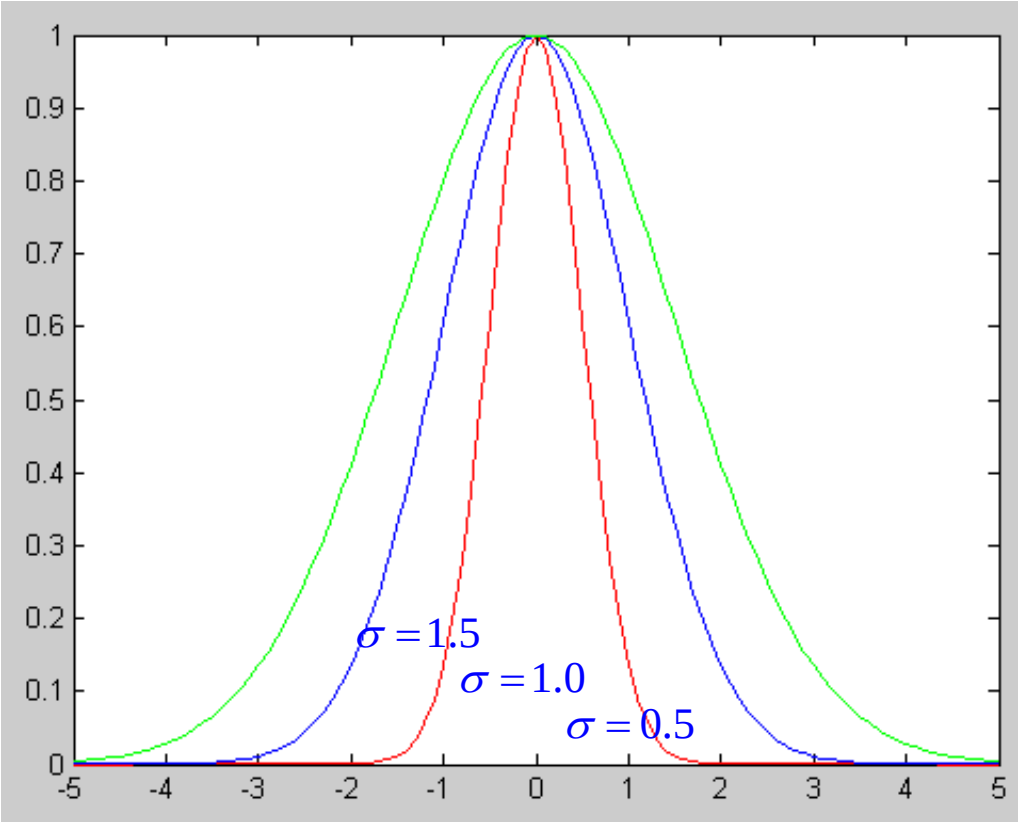
$$\phi(r) = \sqrt{r^2 + c^2} \Big/ c \quad c > 0 \quad \text{and} \quad r \in \Re$$

- **Inverse Multiquadratics:**

$$\phi(r) = c \Big/ \sqrt{r^2 + c^2} \quad c > 0 \quad \text{and} \quad r \in \Re$$

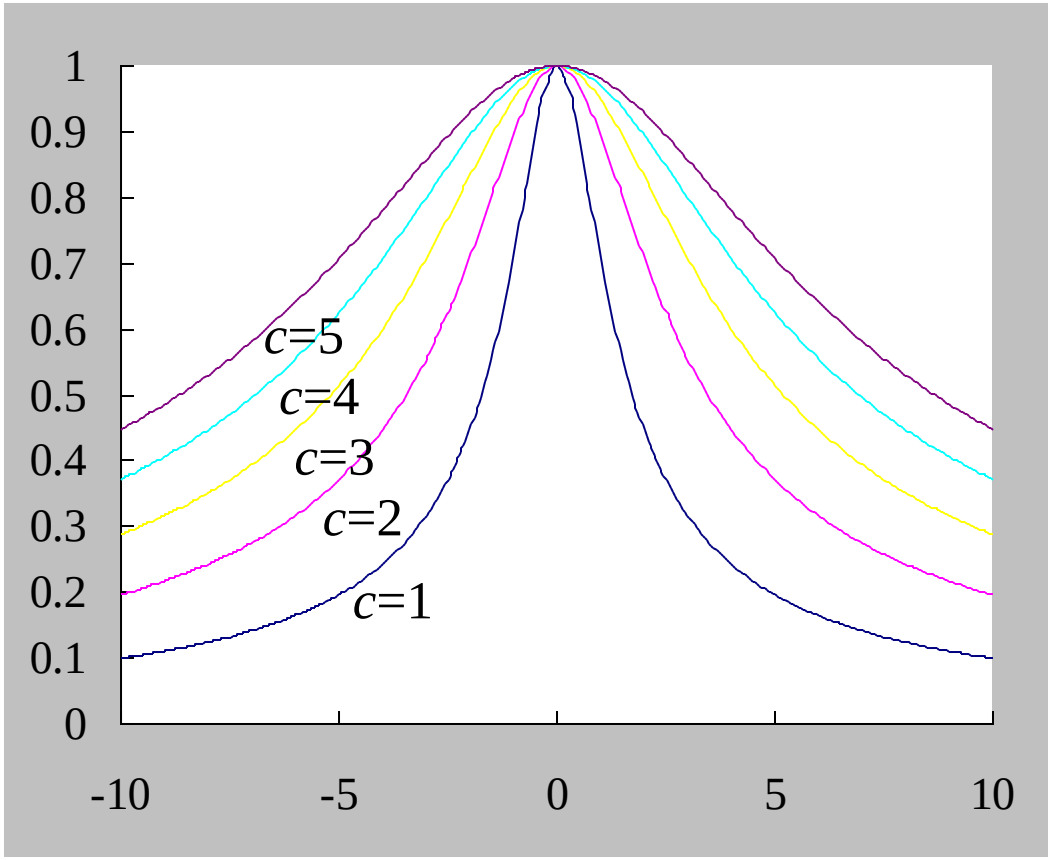
Gaussian Basis Function

$$\phi(r) = e^{-\frac{r^2}{2\sigma^2}} \quad \sigma > 0 \quad \text{and} \quad r \in \Re$$



Inverse Multiquadratics

$$\phi(r) = c \Big/ \sqrt{r^2 + c^2} \qquad c > 0 \quad \text{and} \quad r \in \Re$$



Cover's Theorem

- Consider the use of a RBF network for a complex pattern classification task.
- The problem is basically solved by transforming it into a high-dimensional space in a nonlinear manner.
- Justification: *Cover's theorem* on the *separability of patterns:*
- *A complex pattern classification problem cast in a high-dimensional space nonlinearly is more likely to be linearly separable than in a low-dimensional space.*
- If the patterns are linearly separable, the classification problem is fairly easy to solve.

Cover's Theorem –2

- Consider a family of surfaces.
- Each surface divides an input space into two regions.
- $\mathcal{H} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ is a set of N pattern vectors.
- Each pattern vector belongs to one of the two classes $\mathcal{H}_1$ or $\mathcal{H}_2$.
- This kind of binary partition is called a *dichotomy*.
- A dichotomy is called separable with respect to a family of surfaces if there exists a surface separating the points in class $\mathcal{H}_1$ from those in class $\mathcal{H}_2$.
- Let $\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \ldots, \varphi_{m_1}(\mathbf{x})$ be a set of m_1 real-valued functions.
- Using these functions, we can define for each pattern $\mathbf{x} \in \mathcal{H}$ the vector

$$\mathbf{f}(\mathbf{x}) = \quad [\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \ldots, \varphi_{m_1}(\mathbf{x})]^T$$

Cover's Theorem -3

- Assume now that $\mathbf{x}$ is a m_0-dimensional vector.
- Then the function $\mathbf{f}(\mathbf{x})$ maps points in m_0-dimensional input space into corresponding points in a new space of dimension m_1.
- $\varphi_i(\mathbf{x})$ is referred to as a *hidden function*.
- The space spanned by the functions $\varphi_1(\mathbf{x}), \ldots, \varphi_{m_1}(\mathbf{x})$ is called the *hidden space* or *feature space*.
- The hidden functions have a similar role as hidden units in a MLP network.

Cover's Theorem -4

- A dichotomy $[\mathcal{H}_1, \mathcal{H}_2]$ of $\mathcal{H}$ is said to be $\varphi - separable$ if there exists an m_1-dimensional vector $\mathbf{w}$ satisfying the condition

$$\begin{aligned} \mathbf{w}^T \mathbf{f}(\mathbf{x}) &> 0, \quad \mathbf{x} \in \mathcal{H}_1 \\ \mathbf{w}^T \mathbf{f}(\mathbf{x}) &< 0, \quad \mathbf{x} \in \mathcal{H}_2 \end{aligned}$$

- The hyperplane defined by the equation

$$\mathbf{w}^T \mathbf{f}(\mathbf{x}) = 0$$

  describes the separating surface in the hidden φ-space.

- The inverse image of this subspace, that is,

$$\mathbf{x}: \quad \mathbf{w}^T \mathbf{f}(\mathbf{x}) = 0$$

  defines the *separating surface* in the input space.

Cover's Theorem -5

- Summarizing, Cover's theorem on the separability of patterns has two basic ingredients:
  1. Nonlinear formulation of the hidden functions $\varphi_i(\mathbf{x})$, $i = 1, 2, \ldots, m_1$.
  2. High dimensionality of the hidden space compared to the input space.
- Sometimes the use of nonlinear mapping alone without increasing the dimensionality is sufficient for producing linear separability.

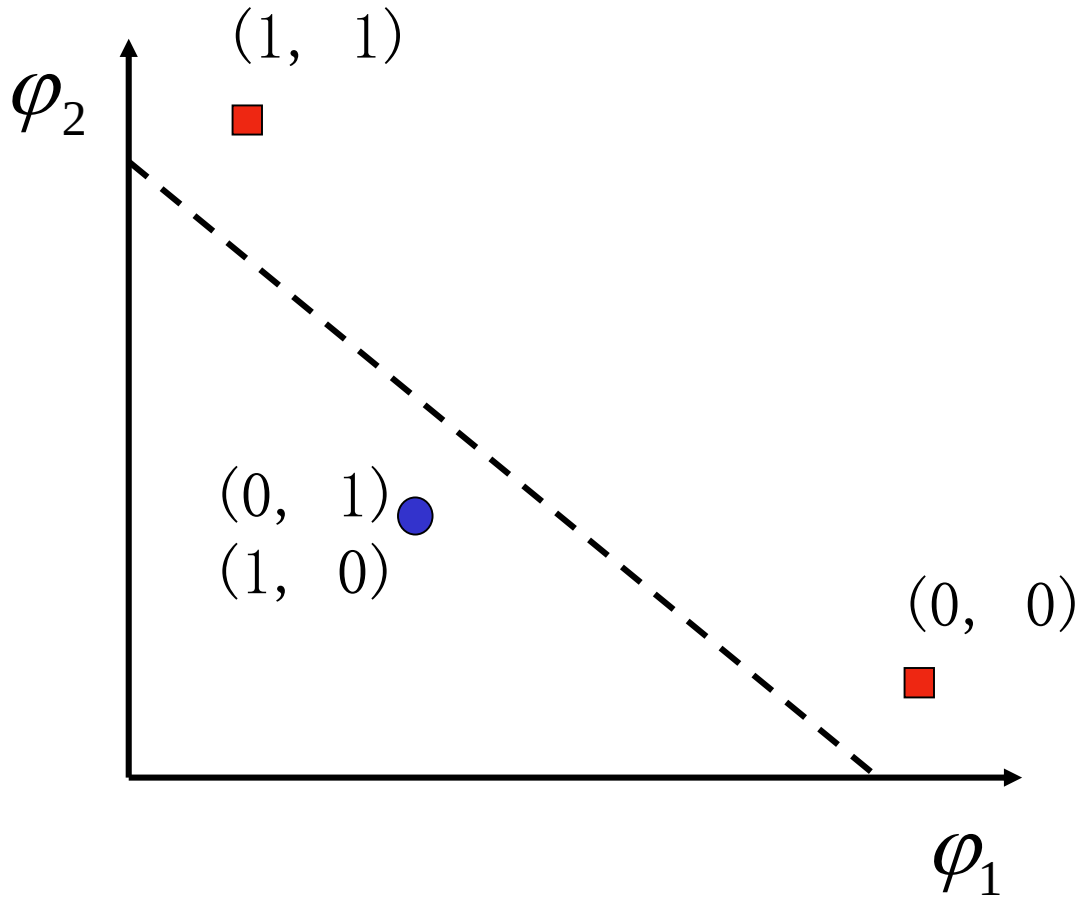
XOR Problem

- For illustrating the importance of φ-separability, consider again the simple yet important XOR problem.
- Four points (patterns) (1,1), (0,1), (0,0), and (1,0) in a two-dimensional input space .
- Requirement: construct a binary classifier with output:
  - 0 for the inputs (1,1) or (0,0)
  - 1 for the inputs (1,0) or (0,1).
- Recall that the XOR problem is not linearly separable in the original input space.
- Define a pair of Gaussian hidden functions

$$\varphi_1(\mathbf{x}) = \exp(-\parallel \mathbf{x} - \mathbf{t}_1 \parallel^2), \qquad \mathbf{t}_1 = [1, 1]^T$$

$$\varphi_2(\mathbf{x}) = \exp(-\parallel \mathbf{x} - \mathbf{t}_2 \parallel^2), \qquad \mathbf{t}_2 = [0, 0]^T$$

Solution of the XOR Problem



Approximation Properties

- Multilayer perceptrons have the universal approximation property.
- Also the family of RBF networks can uniformly approximate any continuous function on a compact set.
- Formally, let $G : \mathcal{R}^n \rightarrow \mathcal{R}$ be integrable, continuous, and bounded function satisfying the condition

$$\int_{\mathcal{R}^n} G(\mathbf{x}) d\mathbf{x} \neq 0$$

- Let $\mathcal{F}_G$ denote the family of RBF networks consisting of functions $F : \mathcal{R}^n \rightarrow \mathcal{R}$

$$F(\mathbf{x}) = \sum_{i=1}^{m_1} w_i G\left(\frac{\mathbf{x} - \mathbf{t}_i}{\sigma}\right)$$

- Here $\sigma > 0$, $w_i \in \mathcal{R}$ and $\mathbf{t}_i \in \mathcal{R}^n$ for $i = 1, 2, \ldots, m_1$.

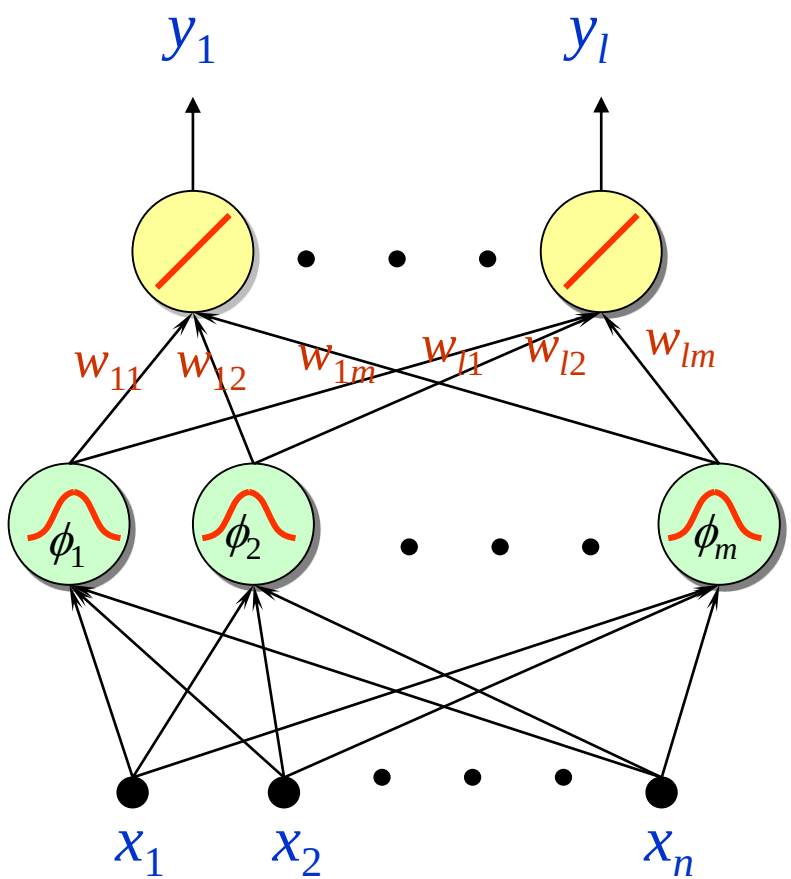
Approximation Properties –2

- *The universal approximation theorem* for RBF networks:
- *For any continuous input-output mapping function $f(\mathbf{x})$ there is an RBF network with*
- *a set of centers $\mathbf{t}_i$, $i = 1, 2, \ldots, m_1$ and a common width σ such that*
- *the input-output mapping function $F(\mathbf{x})$ realized by the RBF network is close to $f(\mathbf{x})$ in the L_p norm, $p \in [1, \infty)$.*
- Note that the kernel $G : \mathcal{R}^n \rightarrow \mathcal{R}$ need not be radially symmetric.
- The theorem provides a theoretical basis for using RBF networks in practical applications.

Learning Strategies

- In RBF networks, learning proceeds differently for different layers.
- The linear output layer's weights are learned rapidly through a *linear* optimization strategy.
- The hidden layer's activation functions evolve slowly using some *nonlinear* optimization strategy.
- The layers of a RBF network perform different tasks.
- It is reasonable to use different optimization techniques for the hidden and output layers.
- Learning strategies for the RBF networks differ in the method used for specifying the centers of the RBF network.

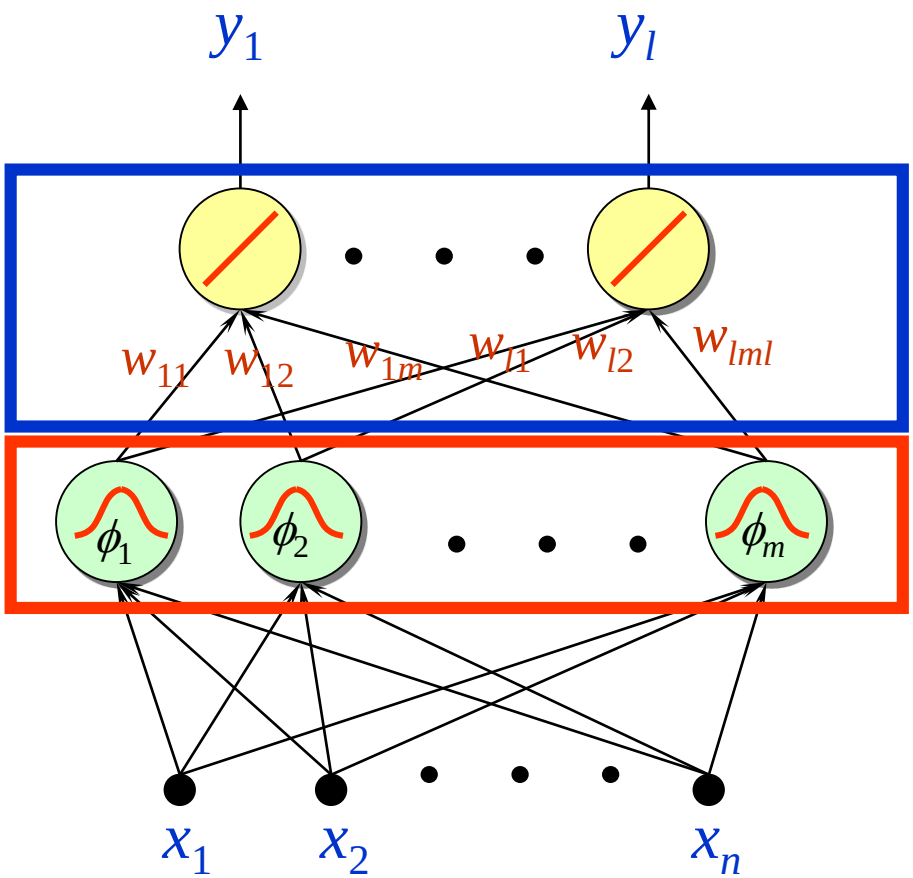
What to Learn?



- Weights: w_{ij}'s
- Centers: μ_j's of ϕ_j's
- Widths: σ_j's of ϕ_j's

- Number of ϕ_j's – Model Selection

Two-Stage Training



Step 2

Determines w_{ij}'s.

E.g., using batch-learning.

Step 1

Determines

- Centers μ_j's of ϕ_j's.
- Widths σ_j's of ϕ_j's.
- Number of ϕ_j's.

Fixed Centers Selected at Random

- The simplest approach is to assume *fixed* radial-basis functions.
- The locations of the centers may be chosen *randomly* from the training data set.
- This is a sensible approach provided that the training data are representative for the problem.
- The radial-basis functions are typically chosen to be *isotropic* Gaussian functions:（各项同性的Gauss函数）

$$G(\| \mathbf{x} - \mathbf{t}_i \|) = \exp\left(-\frac{m_1}{d_{max}^2} \| \mathbf{x} - \mathbf{t}_i \|^2\right)$$

$i = 1, 2, \ldots, m_1$ where m_1 is the number of centers (basis functions).

Comparison of RBF with MLP

- Both RBF and MLP networks are nonlinear layered networks having universal approximation properties.
- The most important differences between them are:
  1. An RBF network has a single hidden layer, while an MLP can have several hidden layers.
  2. The computational nodes in the MLP network are similar in various layers, while in the RBF network they are quite different in the output and hidden layers.

Comparison of RBF with MLP –2

3. In the RBF network, the output layer is linear, while it is usually nonlinear in an MLP network.
4. In each hidden node, the activation function of RBF network computes an *Euclidean distance*, while in MLP networks an *inner product* between the input and the weight vector is computed.
5. MLPs construct *global* approximations, while RBF networks approximate *locally* nonlinear input-output mappings.

- MLP may require less parameters than the RBF network for achieving the same accuracy.