Dendritic Computing in the Lattice Domain

Gerhard X Ritter

CISE Department, University of Florida, USA

Overview

Part I: Background

- Rationale For Lattice Based Dendritic Computing
- Lattice Neural Networks (LNNs)
- Problems Associated With Current LNNs

Part II: A Novel Two Metric Model

- The Basic Idea Behind The Two Metric Model
- Lattice Metrics and Hyperplanes
- The Two Metric Model and Examples
- Concluding Remarks and Questions

Rationale for Dendritic Computing

- Basic Goal: A return of ANNs to its Roots in Neurobiology and Neurophysics
- Radial Basis Function NNs, SVM, Boltzmann Machines, etc., bear little resemblance to biological neural networks
- Dendrites make up more than 50% of a neuron's membrane
- Dendrites make up the largest component in both surface area and volume of the brain
- A neuron in the cortex typically sends messages to approximately 10^4 other neurons.

Rationale for Dendritic Computing

- Dendrites and dendritic spines are major postsynaptic targets of presynaptic inputs
- The number of synapses on a single neuron ranges between 500 and 200,000
- The number of synapses in the human brain ranges between 60 trillion and 240 trillion (240×10^{12})
- These synapses reside on 10 to 20 billion neurons

Rationale for Dendritic Computing

- Recent research results demonstrate that the dynamic interaction of inputs in dendrites containing voltage-sensitive ion channels make them capable of realizing nonlinear interactions, logical operations, and possibly other local domain computation (Poggio, Koch, Shepherd, Rall, Segev, Perkel, et.al.)
- Based on their experimentations, these researchers make the case that it is the *dendrites* and not the neural cell bodies that *are the basic computational units of the brain.*
- Thus, when attempting to model artificial brain networks, one cannot ignore dendrites

Rationale for Lattice Computing

- Neurons with dendrites can function as many *independent subunits* with each unit being able to implement a rich repertoire of *logical operations*
- Logical functions such as **XOR**, **AND**, **OR**, and **NOT**; Koch, Riesenhuber, Poggio, Setiono, Segev and others
- Lattce operations involve only *max*, *min*, and *addition*; i.e., $\vee$, $\wedge$, and $+$
- Thus, lattice operations provide for extremely fast neural computation and fast learning methods

Biological Neurons

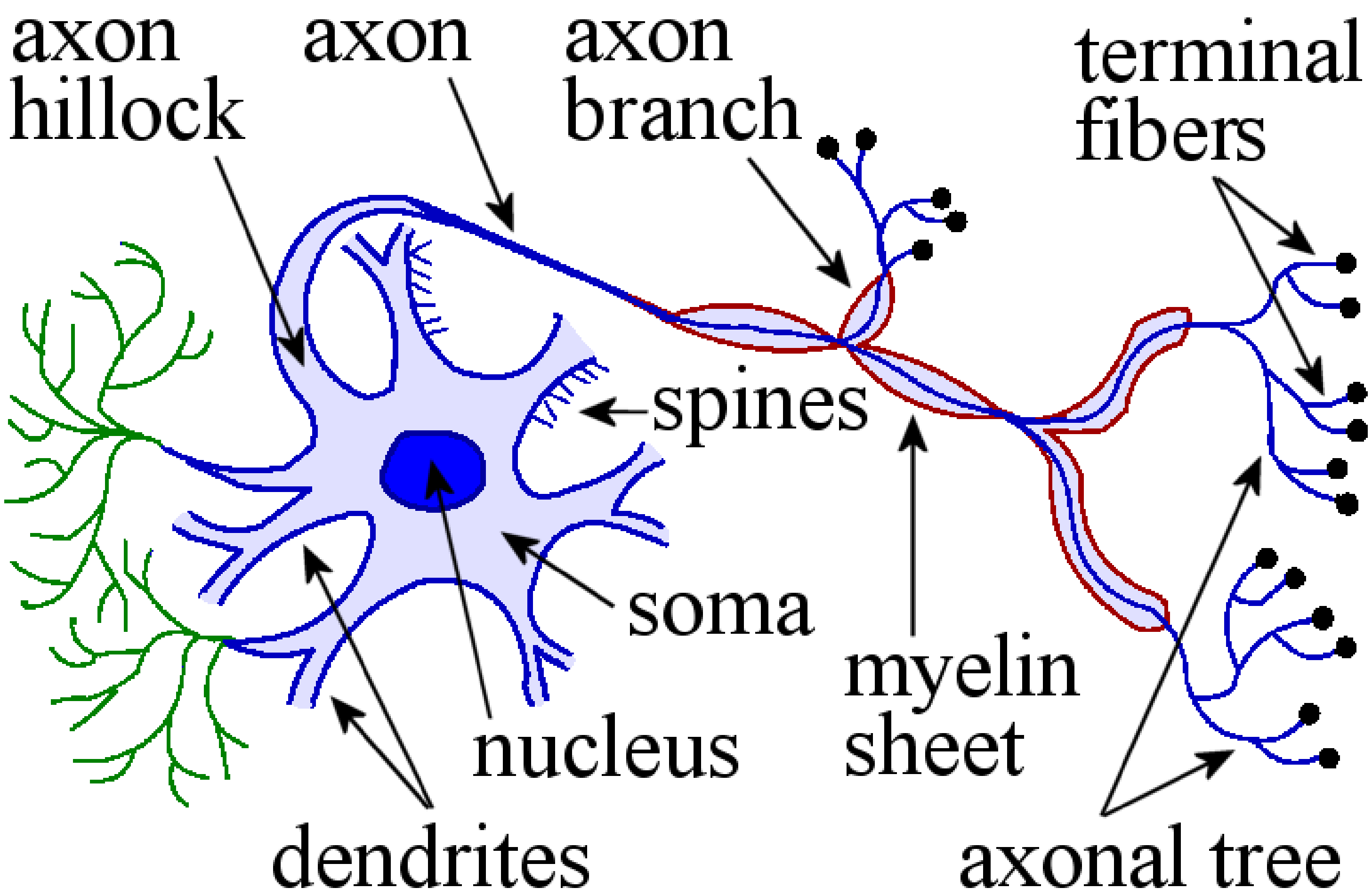


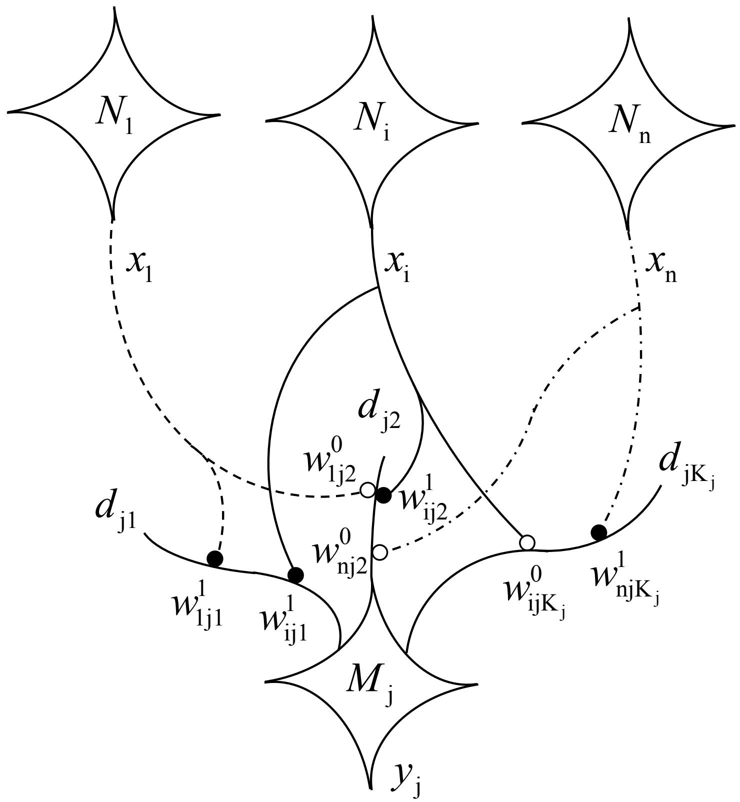
Figure 1: Simplified sketch of the processes of a biological neuron.

Dendritic Computation: Assumptions

- The postsynaptic neuron M_j receives input from n presynaptic neurons $N_1, \ldots, N_n$.
- Each input neuron N_i has axonal branches that terminate at various synaptic regions of M_j.
- The synaptic regions are distributed along a finite number of dendrites $d_1, \ldots, d_{K(j)}$.
- Incoming information from axonal branches is transformed in the synaptic interaction
- The transformed data will result in either an *excitatory* postsynaptic response or an *inhibitory* postsynaptic response in the dendrites membrane.

UNIVERSITY OF FLORIDA

Postsynaptic neuron with dendritic structures



Terminal branches of axonal fibers originating from the presynaptic neurons make contact with synaptic sites on dendritic branches of M_j

Dendritic Computation: Mathematical Model

The computation performed by the kth dendrite for input $\mathbf{x} = (x_1, \ldots, x_n)' \in \mathbb{R}^n$ is given by

$$\tau_k^j(\mathbf{x}) = p_{jk} \bigwedge_{i \in I(k)} \bigwedge_{\ell \in L(i)} (-1)^{1-\ell} \left(x_i + w_{ijk}^{\ell}\right) \; ,$$

where

- x_i – value of neuron N_i;
- $I(k) \subseteq \{1, \ldots, n\}$ – set of all input neurons with terminal fibers that synapse on dendrite d_{jk};
- $L(i) \subseteq \{0, 1\}$ – set of terminal fibers of N_i that synapse on dendrite d_{jk};
- $p_{jk} \in \{-1, 1\}$ – EPSP/IPSP of d_{jk}.

Dendritic Computation: Mathematical Model

- The value $\tau_k^j(\mathbf{x})$ is passed to the cell body and the state of M_j is a function of the input received from all its dendritic postsynaptic results. The total value received by M_j is given by

$$\tau^j(\mathbf{x}) = p_j \bigwedge_{k=1}^{K(j)} \tau_k^j(\mathbf{x}).$$

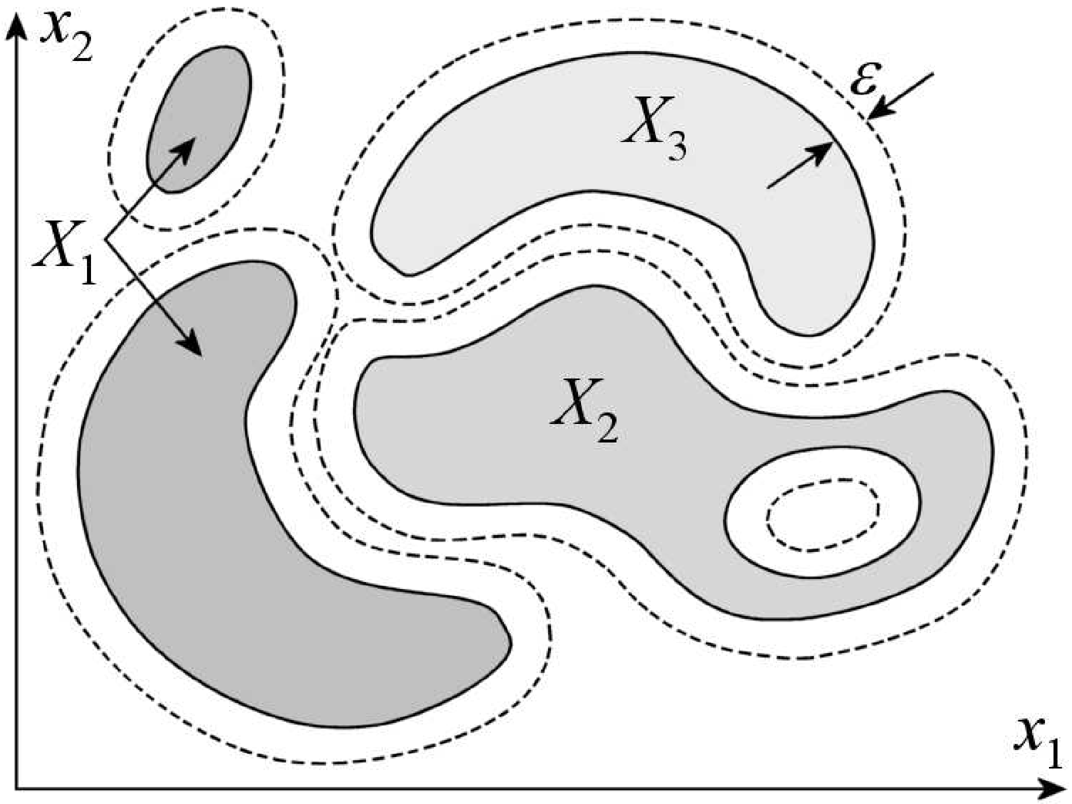
The Capabilities of an SLLP

- An SLLP can distiguish between any given number of pattern classes to within any desired degree of $\varepsilon > 0$.
- More precisely, suppose X_1, X_2, $\ldots$, X_m denotes a collection of disjoint compact subsets of $\mathbb{R}^n$.
- For each $p \in \{1, \ldots, m\}$, define
  $Y_p = \bigcup_{j=1, j \neq p}^{m} X_j$
  $\varepsilon_p = \mathrm{d}(X_p, Y_p) > 0$
  $\varepsilon_0 = \frac{1}{2} \min\{\varepsilon_1, \ldots, \varepsilon_m\}$.
- As the following theorem shows, a given pattern $\mathbf{x} \in \mathbb{R}^n$ will be recognised correctly as belonging to class C_p whenever $\mathbf{x} \in X_p$

The Capabilities of an SLLP

- **Theorem.** *If* $\{X_1, X_2, \ldots, X_m\}$ *is a collection of disjoint compact subsets of* $\mathbb{R}^n$ *and* ε *a positive number with* $\varepsilon < \varepsilon_0$, *then there exists a single layer lattice based perceptron that assigns each point* $\mathbf{x} \in \mathbb{R}^n$ *to class* C_j *whenever* $\mathbf{x} \in X_j$ *and* $j \in \{1, \ldots, m\}$, *and to class* $C_0 = \neg \bigcup_{j=1}^{m} C_j$ *whenever* $\mathrm{d}(\mathbf{x}, X_i) > \varepsilon, \forall i = 1, \ldots, m$. *Furthermore, no point* $\mathbf{x} \in \mathbb{R}^n$ *is assigned to more than one class.*

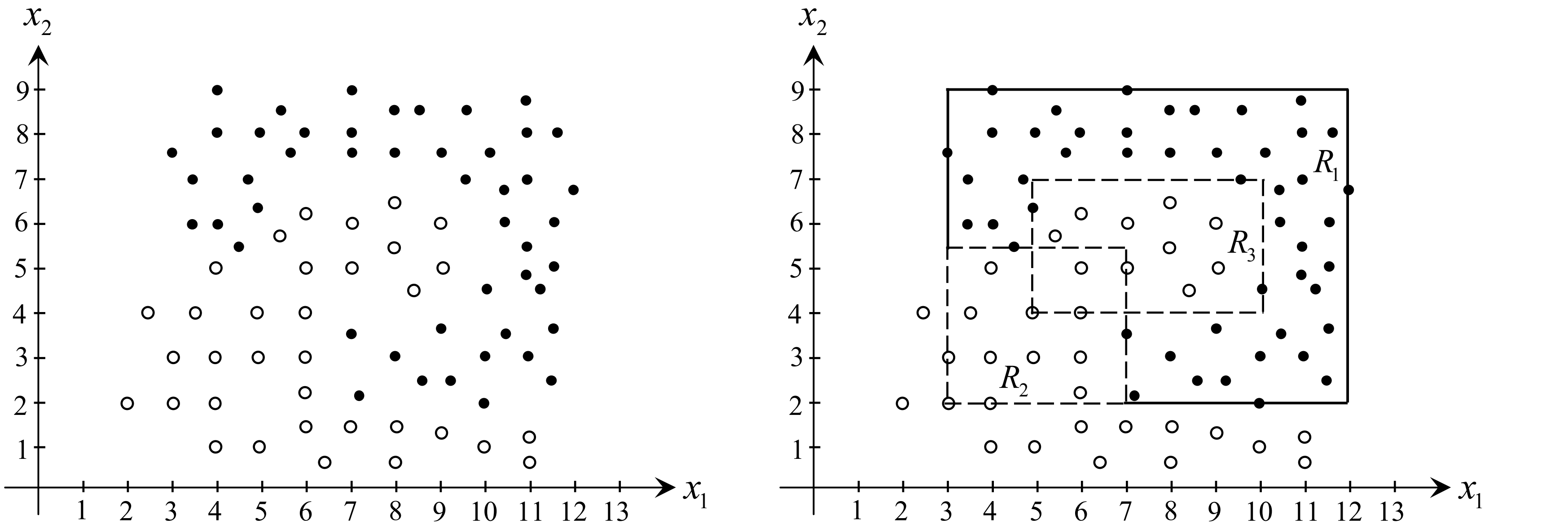
Graphical Interpretation of Theorem



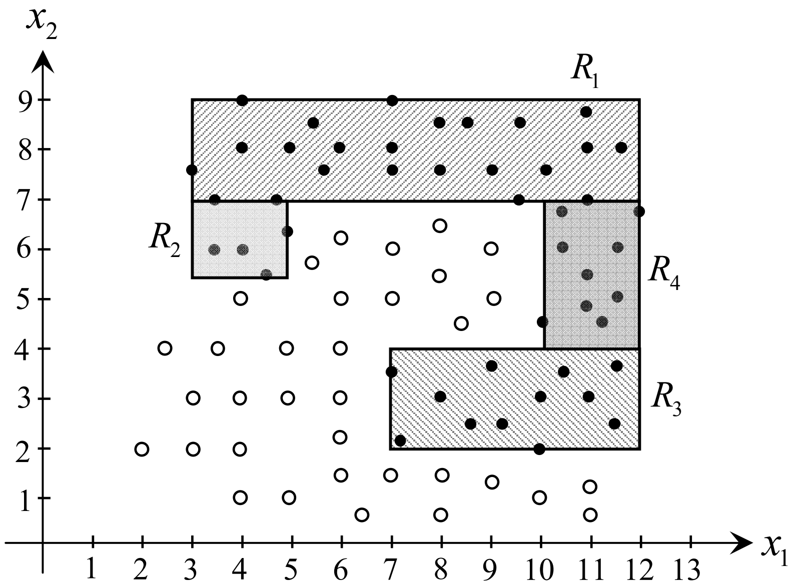
Any point in the set X_j is identified with class C_j; points within the ϵ-band may or may not be classified as belonging to C_j, points outside the ϵ-bands will not be associated with a class C_j $\forall j$.

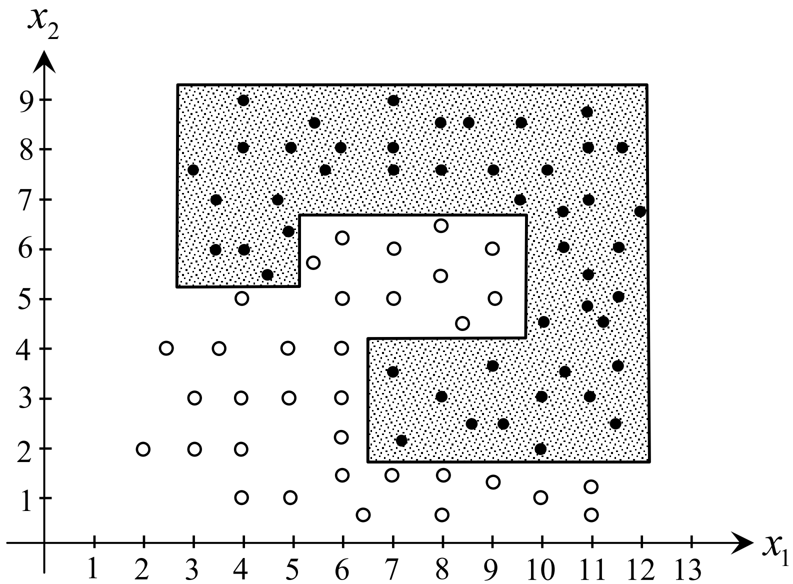
Learning in LNNs

- Early training methods were based on the proofs of the preceding Theorems.
- All training algorithms involve the growth of axonal branches, computation of branch weights, creation of dendrites, and synapses.
- The first training algorithm developed was based on elimination of foreign patterns from a given training set (min or intersection).
- The second training algorithm was based on small region merging (max or union).



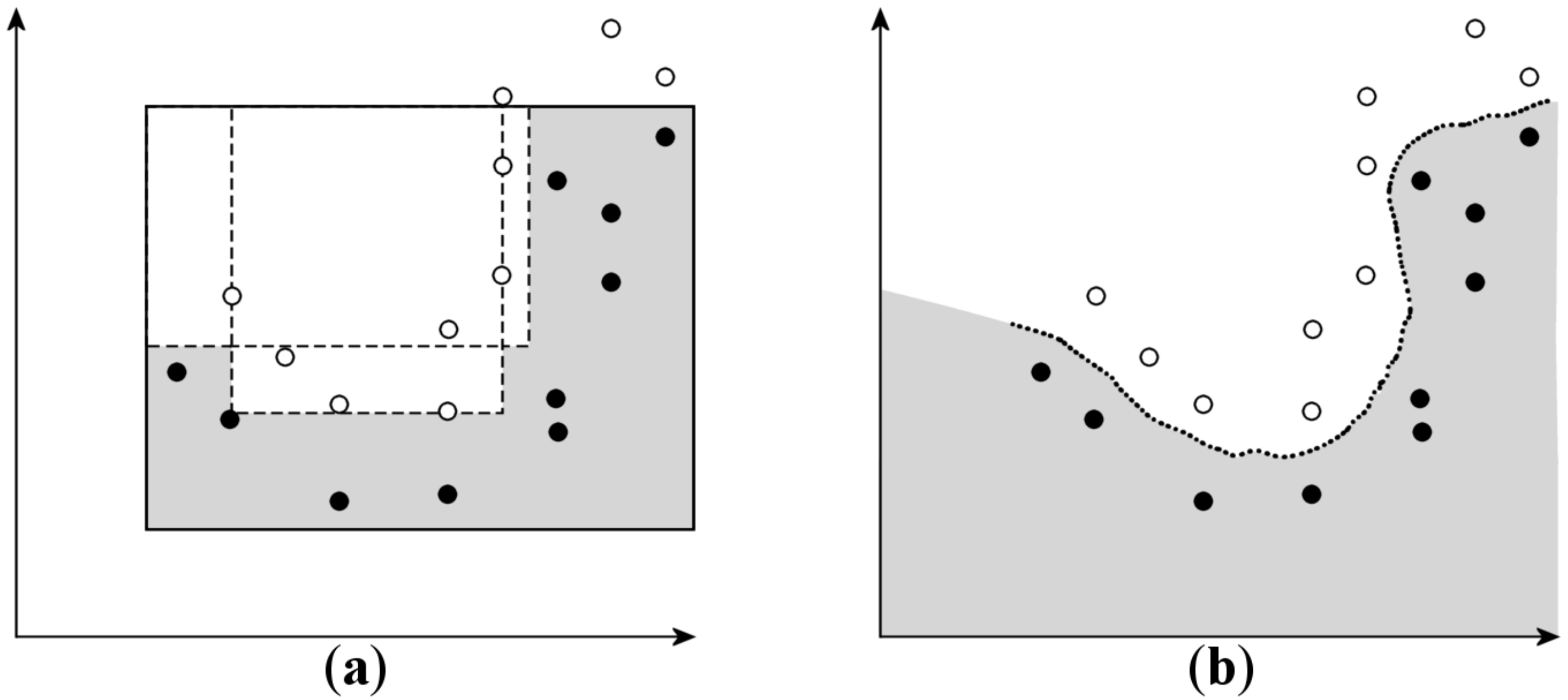
Left : Two class data set. **Right** : The elimination method.





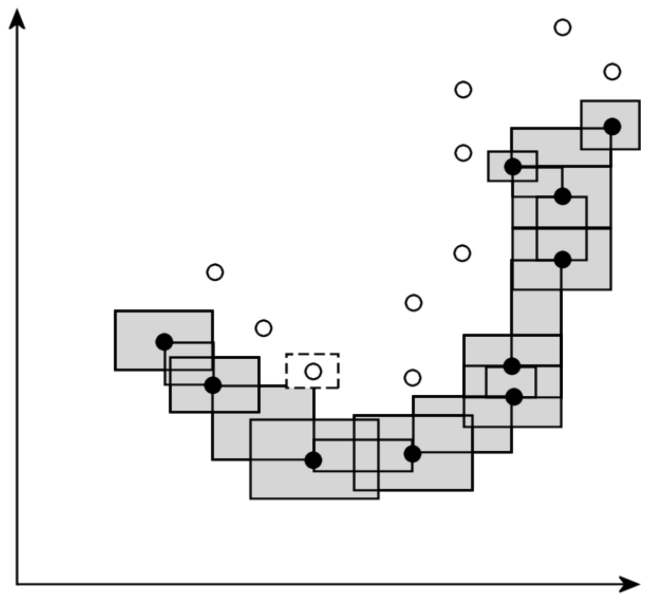
Left : The merging method. **Right** : Boundary readjustment.

SLLP Using Elemination VS MLP



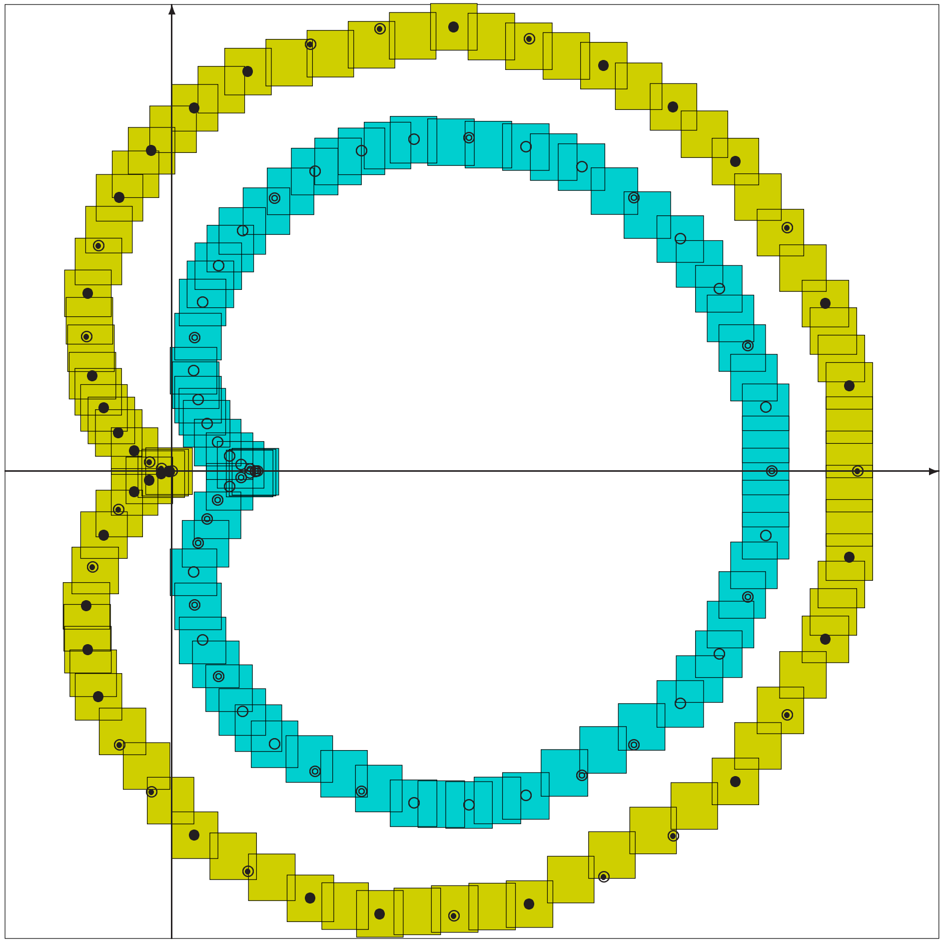
(**a**) SLLP: 3 dendrites, 9 axonal branches. (**b**) MLP 13 hidden neorons and 2000 epochs.

SLLP Using Merging



During training, the SLLP grows 20 dendrites, 19 excitatory and 1 inhibitory (*dashed*).

Another Merging Example



Learning in LNNs

Classifier	Recognition
SLLP (elimination)	98.0%
Backpropagation	96%
Resilient Backpropagation	96.2%
Bayesian Classifier	96.8%
Fuzzy LNN	100%

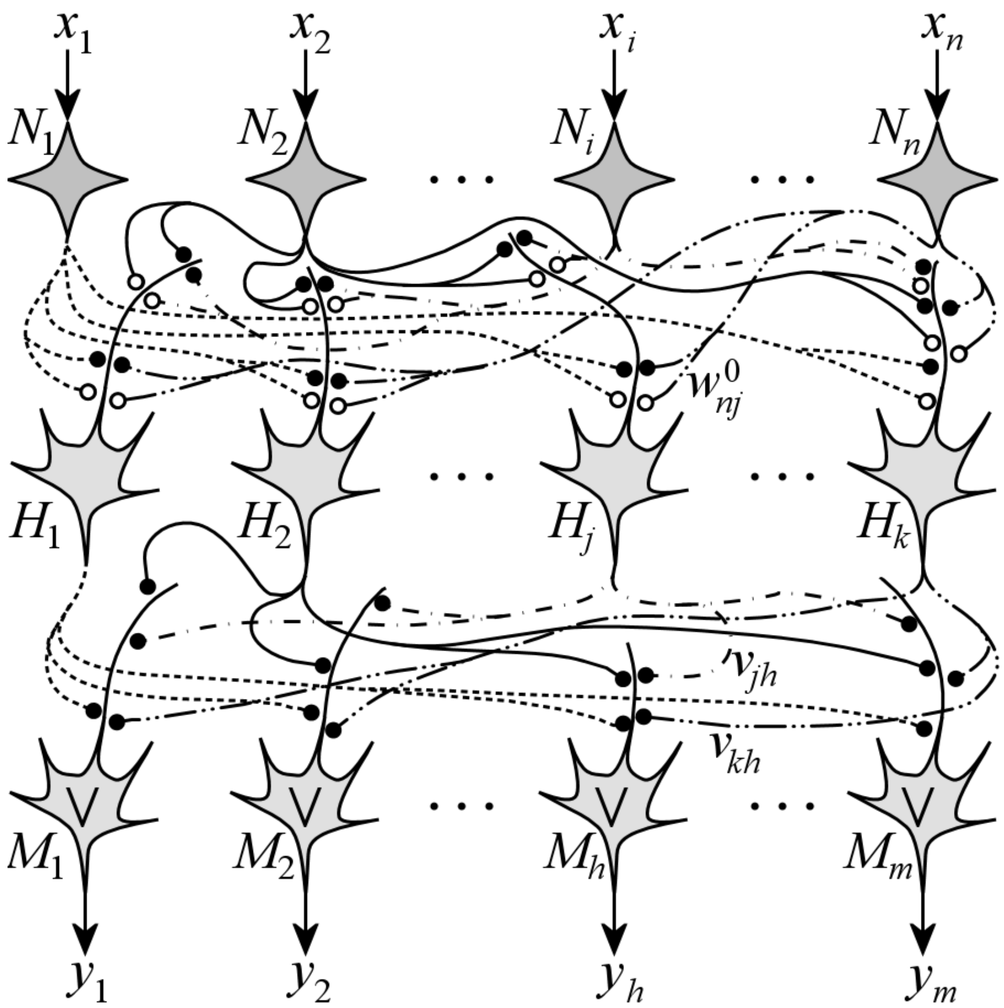
UC Irvine Ionosphere data set (2-class problem in $\mathbb{R}^{34}$ with training set = 65% of data set)

Learning in LNNs

Classifier	Recognition
Fuzzy SLLP (merge/elimination)	98.7%
Backpropagation	95.2%
Fuzzy Min-Max NN	97.3%
Bayesian Classifier	97.3%
Fisher Ratios Discimination	96.0%
Ho-Kashyap	97.3%

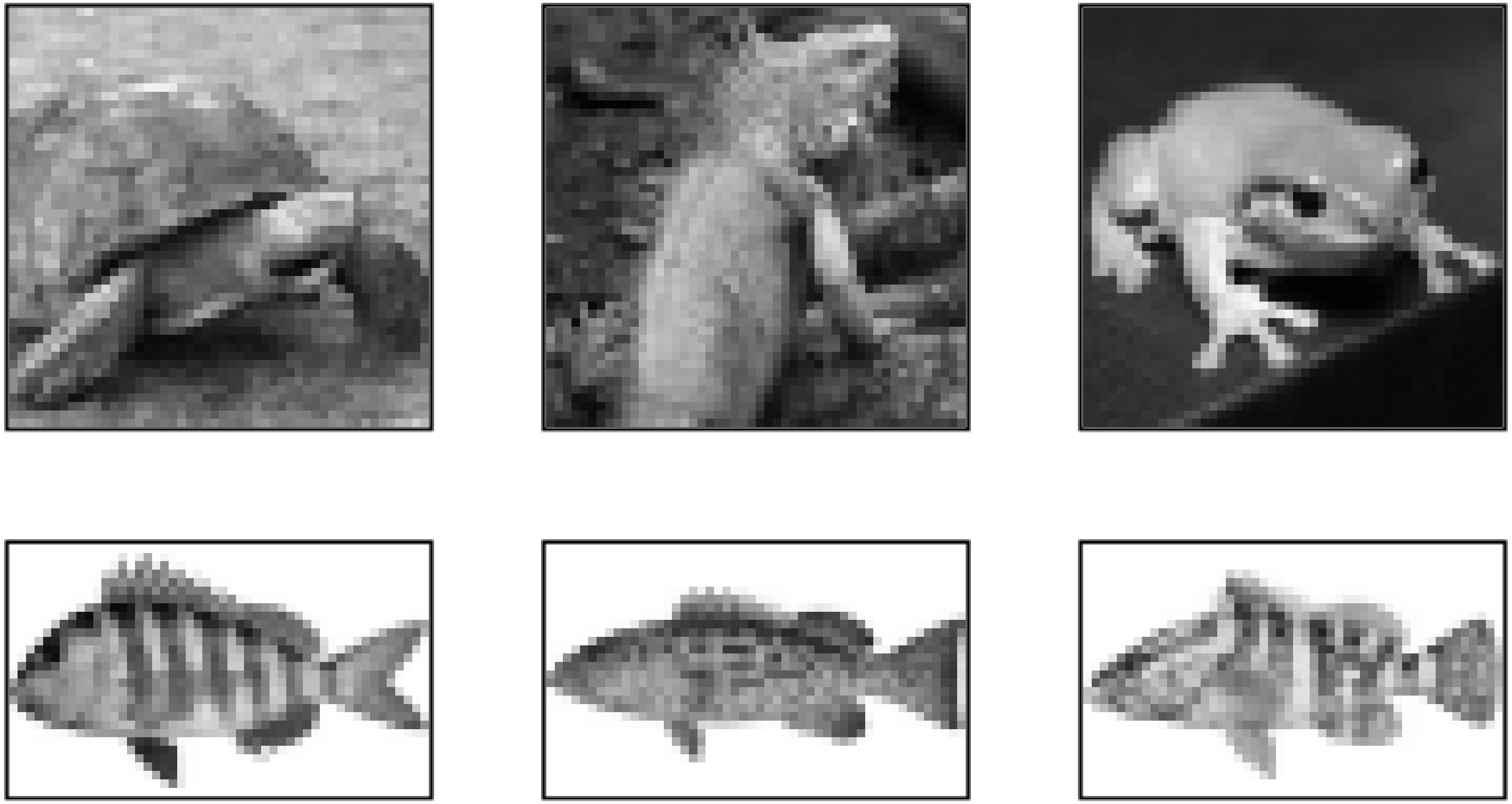
Fisher's Iris Data Set. A 3-class problem in $\mathbb{R}^4$ with training set = 50% of data set.

Dendritic Model of an Associative Memory



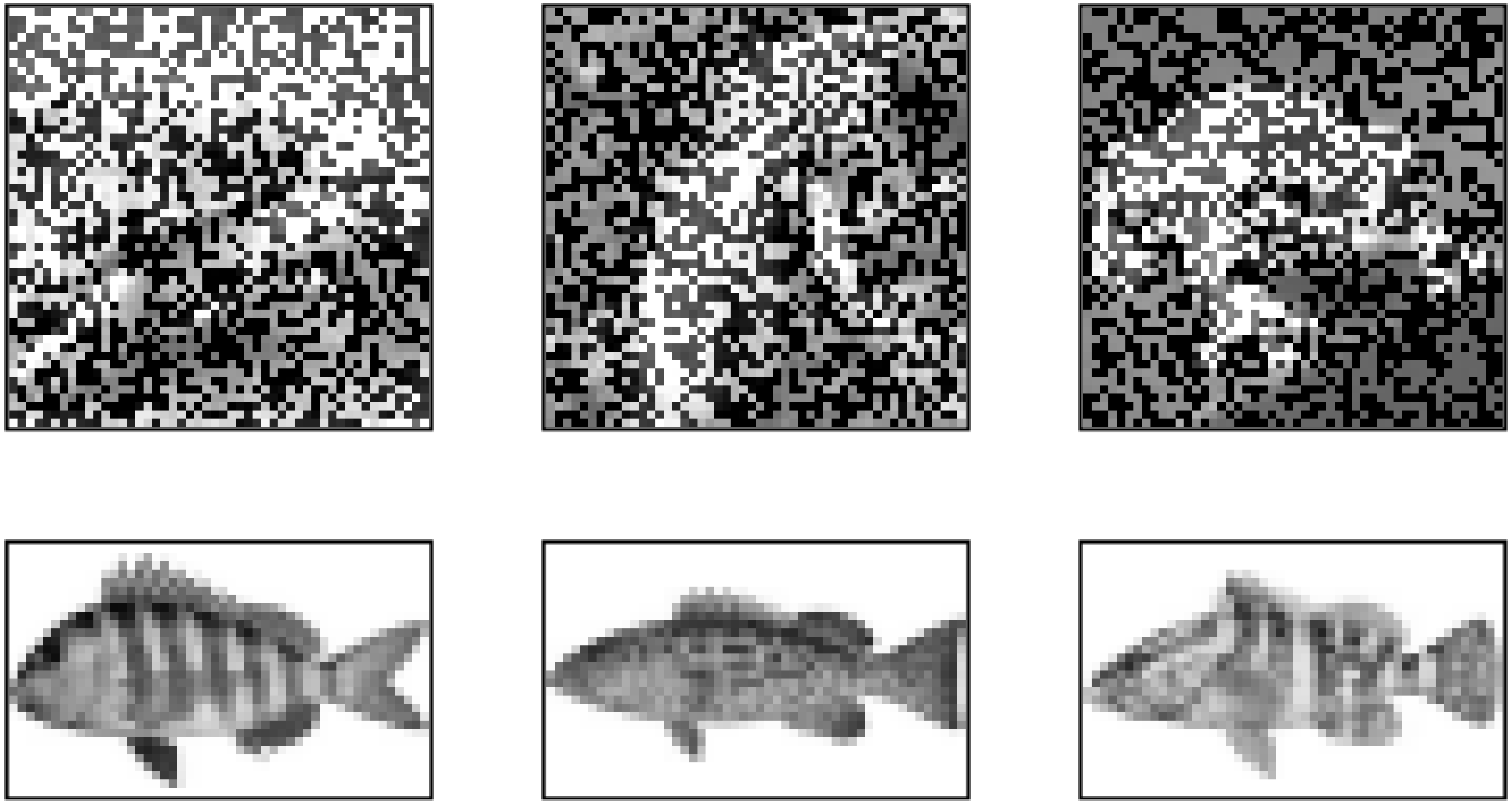
Topology of the dendritic associative memory based on the dendritic model. The network is fully connected.

Patterns to Store



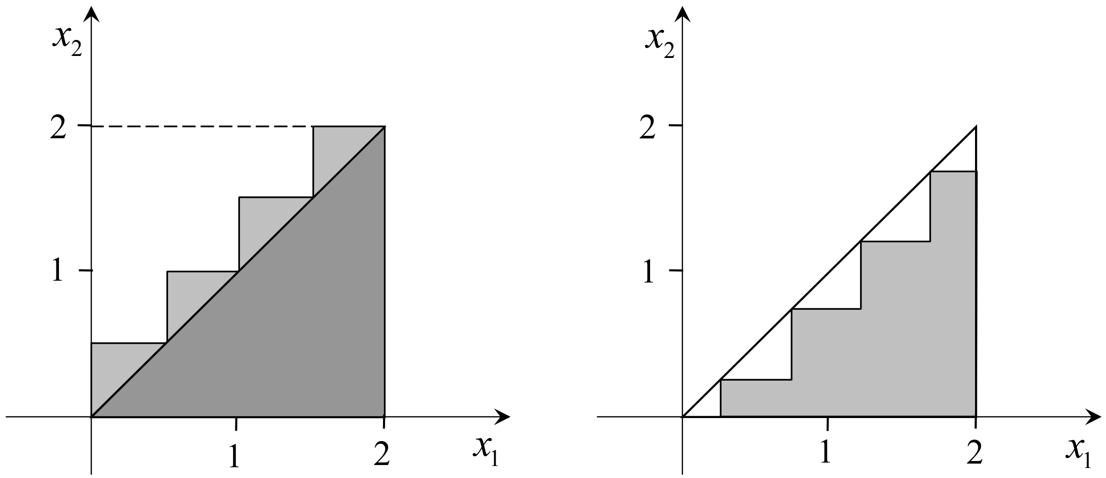
Top row represents the patterns $\mathbf{x}^1, \mathbf{x}^2$, and $\mathbf{x}^3$, while the bottom row depicts the associated patterns $\mathbf{y}^1, \mathbf{y}^2$, and $\mathbf{y}^3$. Here $n = 2500$ and $m = 1500$.

Successful Recall of Noisy Patterns



The top row shows noisy input patterns. Bottom row shows perfect racall association.

Problems with the Hyperbox Approach



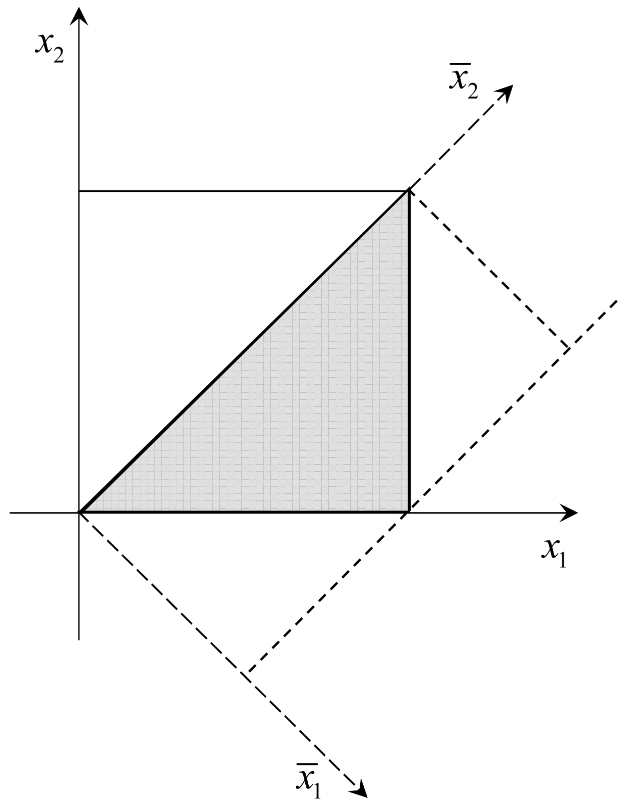
The triangular data can never be modeled *exactly* using either elimination or merging.

Learning in LNNs

- A. Barmpoutis extended the elimination method to arbitrary orthonormal basis settings.
- Basic Equation:

$$\tau_k^j(\mathbf{x}) = p_k^j \bigwedge_{i \in I(k)} \bigwedge_{\ell \in L(i)} (-1)^{1-\ell} \left((\mathbf{R}_k \cdot \mathbf{x})_i + w_{ijk}^{\ell} \right)$$

Problems with Rotations

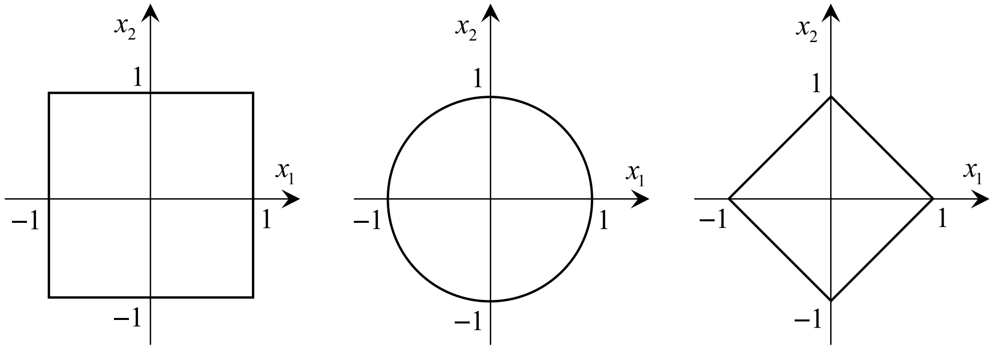


The minimal standard L_∞-rectangle and the minimal 45° OB-rectangle are as shown.

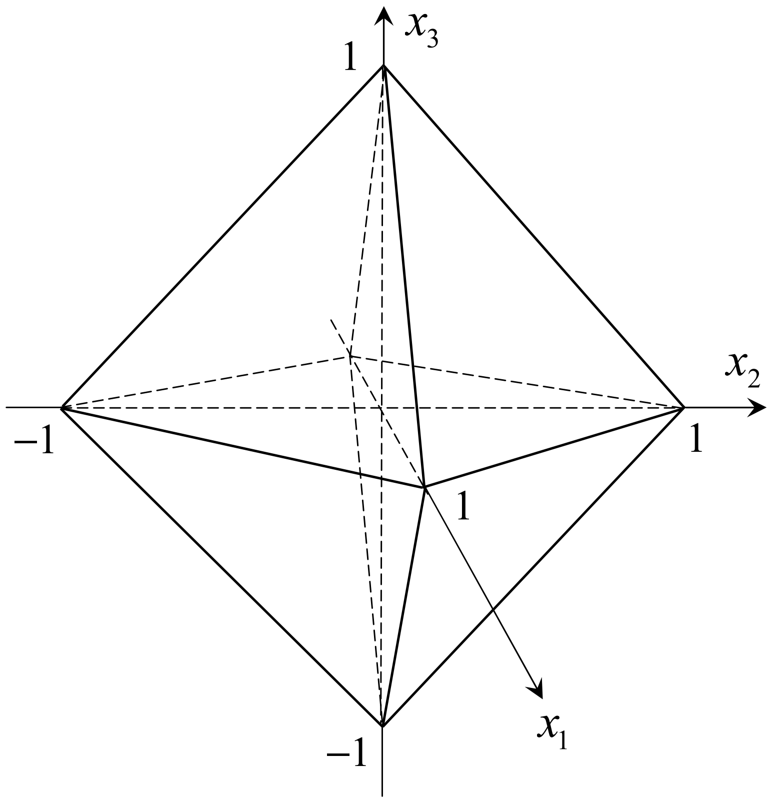
UNIVERSITY OF FLORIDA

Lattice Metrics

- L_1 metric $d_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i|$
- L_∞ metric $d_\infty(\mathbf{x}, \mathbf{y}) = \bigvee_{i=1}^{n} |x_i - y_i|$
- Hausdorff metric based on either the d_1 or d_∞ metric



The d_∞, d_2, and d_1 Spheres



The d_1 Sphere in $\mathbb{R}^3$

Hyperplanes

- A hyperplane in $\mathbb{R}^n$ is defined by

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b,$$

  where not all the a_i's are zero
- If $a_i \in \{-1, 1\} \; \forall \, i$, then the hyperplane is an L_1-hyperplane
- If $a_i = \pm 1$ and $a_j = 0 \; \forall \, j \neq i$, then the hyperplane is an L_∞-hyperplane.

Pertinent Hyperplane Properties

- A hyperplane can also be defined by the function

$$f(\mathbf{x}) = a_1x_1 + a_2x_2 + \cdots + a_nx_n - b = 0$$

- A hyperplane separates $\mathbb{R}^n$ into two half-spaces H^+ (i.e. $f(\mathbf{x}) \geq 0$) and H^- (i.e. $f(\mathbf{x}) \leq 0$)
- Up to parallelism, there are n L_∞-hyperplanes and 2^{n-1} L_1-hyperplanes in $\mathbb{R}^n$

Summation Formulae for L_1-Hyperplanes

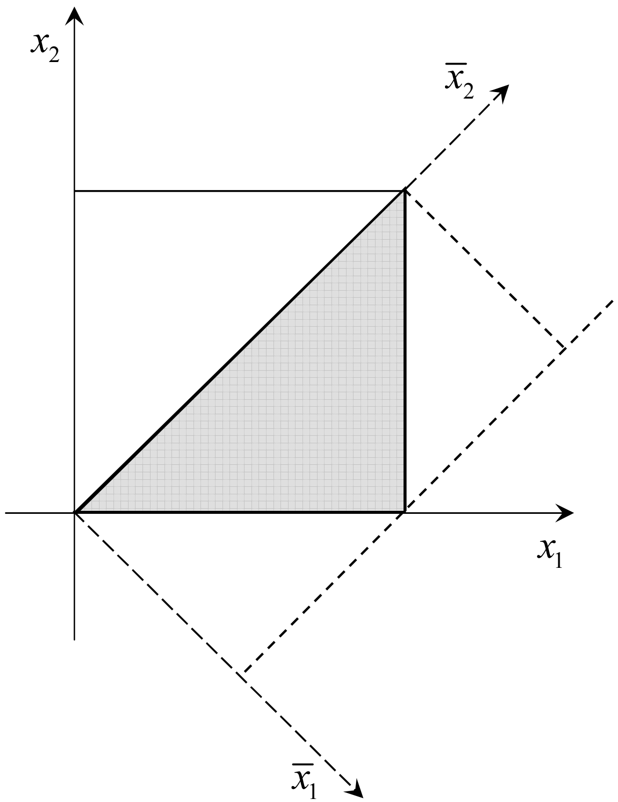
Starting with the summations

$$\begin{aligned} L_1^1(\mathbf{x}) &= x_1 + x_2 \\ L_2^1(\mathbf{x}) &= -x_1 + x_2 \end{aligned}$$

it is easy to generate all summations for a given dimension n using the recursion formula:

$$L_j^{n-1}(\mathbf{x}) = \begin{cases} L_j^{n-2}(\mathbf{x}) + x_n & \text{if} \quad j = 1, \ldots, 2^{n-2} \\ -L_i^{n-2}(\mathbf{x}) + x_n & \text{if} \quad j = 2^{n-2} + i, \end{cases}$$

where $i = 1, \ldots, 2^{n-2}$.



Point $\mathbf{x} = (x_1, x_2)$ is in the triangle $\Leftrightarrow$

$$L_1(\mathbf{x}) \wedge (4 - L_1(\mathbf{x})) \wedge (L_2(\mathbf{x}) + 2) \wedge -L_2(\mathbf{x})$$
$$\wedge x_1 \wedge (2 - x_1) \wedge x_2 \wedge (2 - x_2) \geq 0$$

The Two Metric Model

- The previous inequality is equivalent to

$$\tau(\mathbf{x}) \;=\; \left[\bigwedge_{i=1}^{2}\bigwedge_{\ell=0}^{1}(-1)^{1-\ell}\left(L_i(\mathbf{x})+\omega_i^{\ell}\right)\right]$$

$$\wedge\;\left[\bigwedge_{i=1}^{2}\bigwedge_{\ell=0}^{1}(-1)^{1-\ell}\left(x_i+w_i^{\ell}\right)\right]\geq 0$$

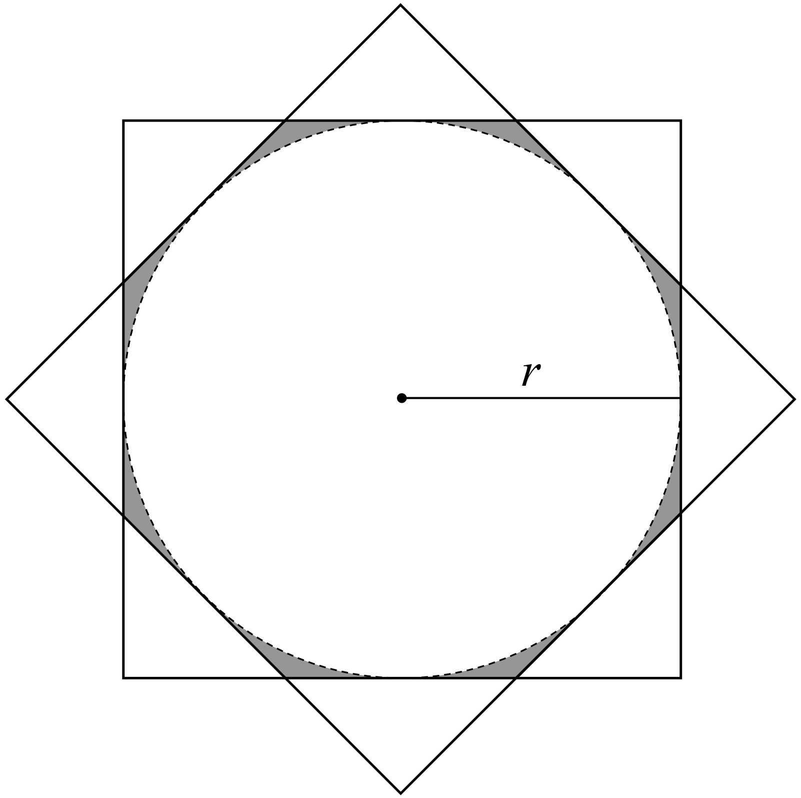
where $\omega_1^1 = \omega_2^0 = w_1^1 = w_2^1 = 0$, $\omega_2^0 = -4$, $\omega_2^1 = 2$, and $w_1^2 = -2 = w_2^0$.

The Two Metric Model

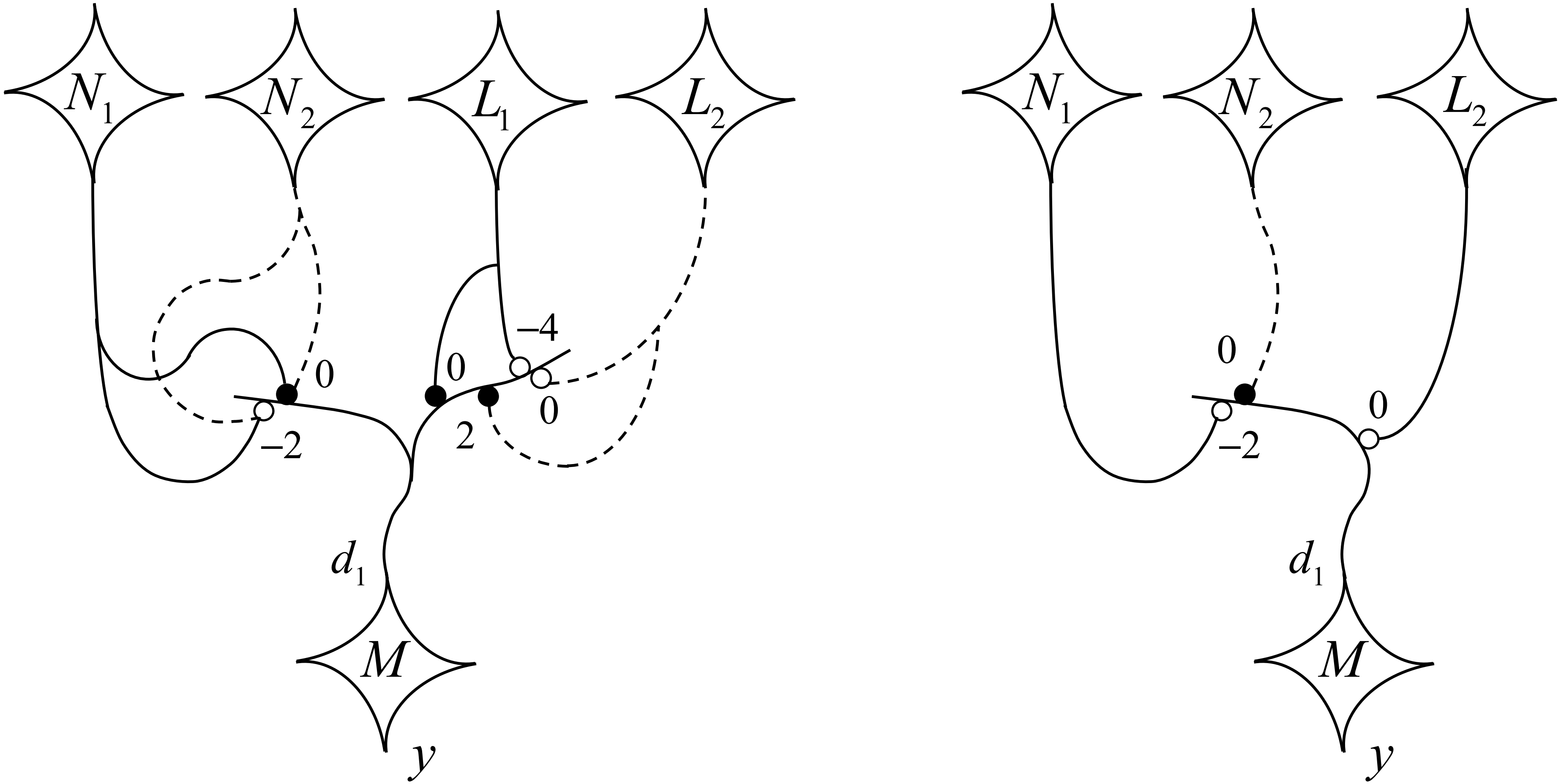
- This generalizes to

$$\begin{aligned}\tau_k^j(\mathbf{x}) &= \Big[p_k^j \bigwedge_{i\in I(k)} \bigwedge_{\ell\in L(i)} (-1)^{1-\ell}(x_i + w_{ijk}^\ell)\Big] \\ &\wedge \Big[q_k^j \bigwedge_{i\in J(k)} \bigwedge_{\ell\in L'(i)} (-1)^{1-\ell}(L_i(\mathbf{x}) + \omega_{ijk}^\ell)\Big],\end{aligned}$$

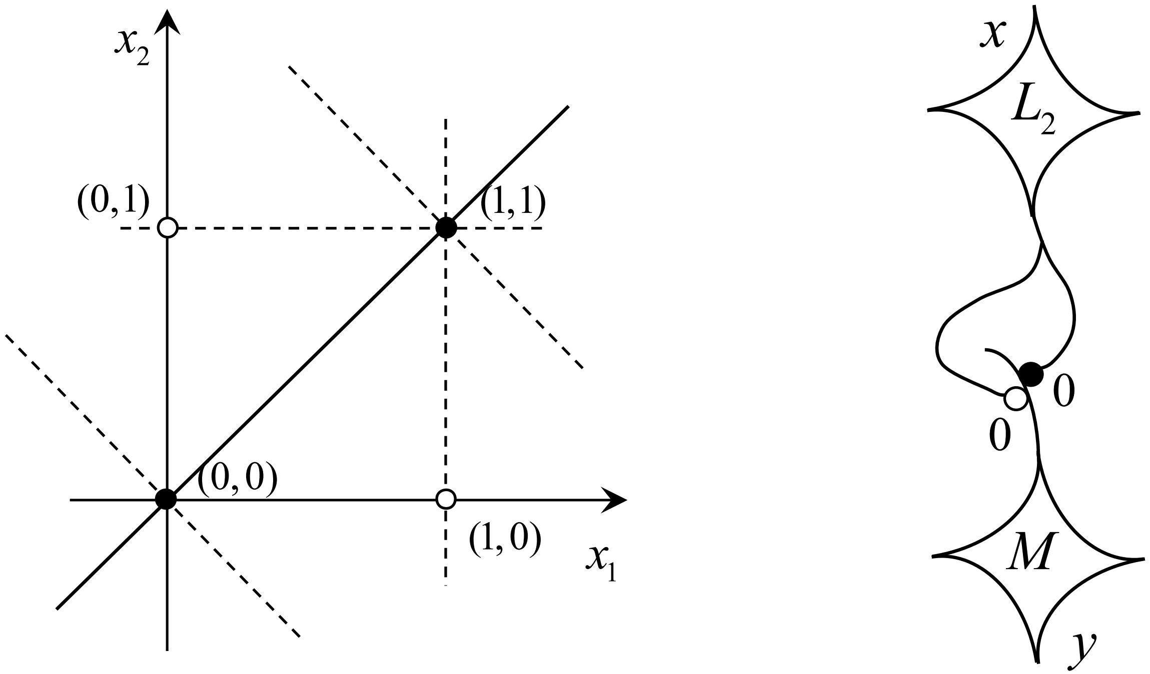
where ω_{hjk}^ℓ is the synaptic weight at synapse of L_i, $J(k) \subseteq \{1, \ldots, 2^{n-1}\}$ set of all input neurons L_i with terminal fibers on d_{jk}.



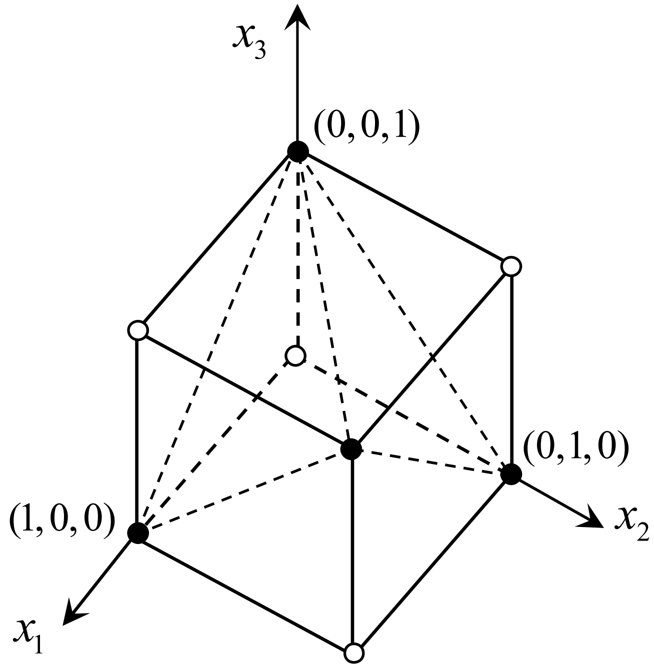
Two metric error: $Area(P^2 \cap H^2) - \pi r^2$, VS single metric error: $Area(H^2) - \pi r^2 = r^2(4 - \pi)$.

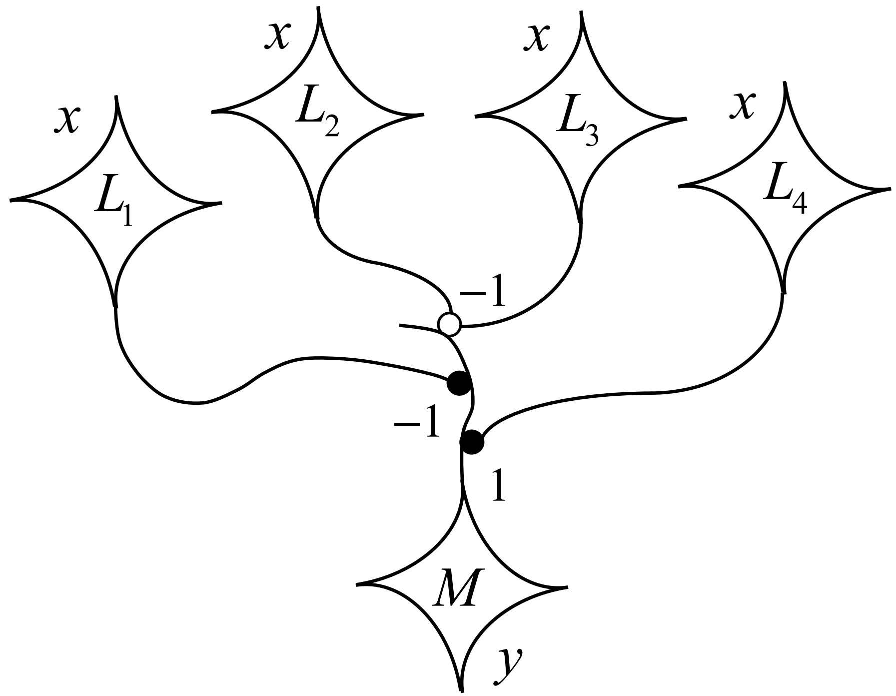


Left : LNN solving the triangle problem derived from learning (elimination). **Right** : LNN after Pruning.



Left : The 2-D XOR Problem. **Right** : LNN derived from learning followed by pruning





Left : The 2-D XOR Problem. **Right** : LNN derived from learning followed by pruning

Questions?

Thank you!