Def of Intelligent Systems: exhibits intelligent behaviors.

Intelligent Behaviors: Inference, e.g., judgment and pattern recognition; Learning and adaptation, e.g., learning from examples; Creativity, e.g., planning and design.

Engineering Applications of Intelligent Systems: Pattern recognition (image processing); Control and robotics (modeling and estimation); Associative memory (content-addressable memory); Forecasting (financial engineering).

Computational Intelligence <- Neural Networks, Fuzzy Logic, evolutionary computing techniques.

Hard computing (quantitative, precise, formal): Bivalent Logic, Numerical analysis, probability theory.

Soft computing (qualitative, imprecise, informal): fuzzy Logic, neurocomputing, genetic computing.

Neural Networks <- artificial neurons, inter-neuron connections, input and output channels.

Formalization of Neural Networks: ANN = (ARCH, RULE)

ARCH: combination of components. RULE: rules relating to the components.

ARCH = (u, v, w, x, y) u and v: neurons. w: inter-neuron connection weights. x and y: external input and outputs.

Connections between Neurons: Adaptive, Excitatory (positive weight) vs. inhibitory (negative weight), Distributed knowledge representation.

RULE = (E, F, G, H, L)

E: Evaluation rule mapped from v and/or y to a real line, e.g., error function or energy function.

F: Activation rule mapped from u to v, e.g., activation function.

G: Aggregation rule mapped from v, w, and/or x to u, e.g., weighted sum.

H: output rule mapped from v to y.

L: Learning rule mapped from v, w, and x to w.

General Incremental Learning Rule: w(t+1) = w(t) + L(v, w, x), ∆w(t) = L(v, w, x)

Two-Time Scale Dynamics in Neural Networks: short-term memory, faster dynamics represented by u and v; long-term memory, slower dynamics represented by w.

Categories of Neural Networks: Deterministic vs. stochastic, in terms of F; Feedforward vs. recurrent, in terms of G and H; Semilinear vs. higher-order, in terms of G; Supervised vs. unsupervised, in terms of L.

Def of Neural Networks: massive parallel distributed processors, storing experiential knowledge.

Features (Resemble brains in two aspects): Knowledge acquisition, learning processors; Knowledge representation: inter-neuron connections.

Properties of Neural Networks: Nonlinearity, Input-output mapping, Adaptivity, Contextual information, Fault tolerance.

Threshold Logic Units: Any logical function F: can be implemented with a two-layer McCulloch-Pitts network; Uninhibited threshold logic units of McCulloch-Pitts type can only implement monotonic logical functions.

Finite Automata: take only a finite set of possible states; react to only a finite set of input signals.

Finite Automata & Recurrent Networks: Any finite automaton can be simulated with a recurrent network of McCulloch-Pitts units.

Perceptron: A single adaptive layer of feedforward network of pure threshold logic units. Simple Perceptron: a computing device with a threshold logic unit.

Linear Separability: the weighted sum, one set threshold, the other set threshold.

Perceptron Convergence Theorem: two sets are linearly separable; the perceptron learning algorithm converge to a set of weights and a threshold in a finite steps.

Limitations of Perceptrons: only linearly separable data can be classified; the convergence rate is low for high-dimensional or large number of data.

Unipolar: Bipolar:

Bipolar coding better than unipolar one: algebraic structure, region proportion in weight space.

ADALINE(improvement of percertron): A single adaptive layer of feedforward network of linear elements. Trained using Delta Rule or LMS Algorithm.

Training Mode of percetron: Sequential mode: input training sample pairs one by one orderly or randomly. Batch mode: input training sample pairs in the whole training set at each iteration.

Perceptron learning: either sequential or batch mode. ADALINE training: batch mode only.

Number of Weight Space Regions:

The learnability problem: when n is large, there is not enough classification regions in weight space to represent all logical functions.

Backpropagation Algorithm: A recursive gradient-descent learning algorithm for multilayer feedforward networks of sigmoid activation function. Compute errors backward from the output layer to input layer. Minimize the mean squares error function.

Process of Backpropagation Algorithm: 1. Initialize weights and threshold randomly; 2. Calculate actual output of the MLP; 3. Adapt weights for all layers

;

4. Repeat until w converges.

Overfitting Problem: model becomes too tailored to the training examples and loses its ability to generalize well to unseen cases.

Function of Momentum Term, to avoid local oscillation.

Multilayer feedforward neural networks: Universal Approximators of continuous functions. A set of weights exist such that the approximation errors can be arbitrarily small. However, the BP algorithm is not guaranteed to find such a set of weights.

Radial Basis Functions: a real-valued function whose value depends only on the distance from its origin or center. Means for approximating or interpolating multivariate functions.

The most commonly used radial-basis function is a Gaussian function.

Radial Basis Function Networks: hidden neurons, a linear combination of a number radial basis functions; Two-layer architecture, output layer uses a linear activation function as ADALINE, hidden layer uses radial basis activation functions.

Cover's Theorem: A dichotomy {X+,X-} is said to be φ-separable if there exist an m-dimensional vector w such that wTφ(x)>0, if x in X+ , wTφ(x) < 0, if x in X-, wTφ(x)=0 is separating surface between the two classes.

An RBF network can transform the linearly inseparable XOR data in the input space to linearly separable data in the hidden state space.

Functional Link Network: One-layer feedforward architecture; Higher-order aggregation rule; Fast learning process; Local minima could be eliminated.

Extreme Learning Machine: One-layer feedforward architecture; One-layer feedforward architecture; Fast learning process for weights in output layer; Local minima eliminated.

Support Vector Machine: Minimization of structural risk; Maximal generalization power.

The linear discriminant function (classifier) with the maximum margin is the best with maximal generalization power. Why: Robust to outliners; strong generalization ability.

What to do if data is not linearly separable? Slack variables can be added to allow misclassification of nonlinearly distributed or noisy data.

Feature Space (Nonlinear SVM): The original input space can be mapped to a higher-dimensional feature space where the training set is linearly separable.

SVM Learning: 1.Choose a kernel function; 2.Choose a value for C; 3.Solve the quadratic programming problem; 4.Construct the discriminant function from the support vectors.

Design Issues: 1. Choice of kernel, Gaussian or polynomial kernel, ineffective->more elaborate kernels; 2.Choice of kernel parameters, σ is the distance between closest points with different classifications, the absence of reliable criteria->use cross-validation; 3.Optimization criterion, Hard margin vs. Soft margin.

The Kernel Trick: No need to know mapping, only use the dot product of feature vectors.

kernel function: corresponds to a dot product of two feature vectors in some expanded feature space.

MAXNET: A sub-network for selecting the input with maximum value - winner takes all.

Principle: mutual inhibition to keep the maximal input and press down the rest.

Usage: the output layer in neural networks.

Network structure: A recurrent neural network with self-excitatory connections and laterally inhibitory connections.

kWTA Model: Compared to MAXNET: MAXNET: Keep a maximum value; high complexity and need to converge. kWTA: Keep the first k maximum values; low complexity.

Heaviside activation function: Desirable Properties: 1. globally stable; 2. globally convergent to the kWTA solutions in finite time. Lower and upper bounds of convergence

time: 

Clustering: Def: group similar data based on a given similarity measure. Features: Subjective without unique solutions; belong to unsupervised learning.

ART1 Network: cluster binary data with/unknown cluster number; A two-layer recurrent neural network; MAXNET serves as its output layer; Bidirectional adaptive connections: bottom-up and top-down connections.

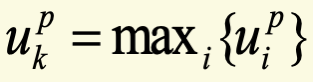
Process for clustering: 1. Initialize weights:



2. Compute net input for an input pattern *x*p:



3. Select the best match using the MAXNET:



4. Vigilance test: If , then next; otherwise, disable neuron *k* and go to step 2).

5. Adapt weights:



Vigilance Parameter in ART1 Network: between 0 and 1; Determine in an ad hoc way; Set granularity of clustering; Define basin of attraction of each cluster. Two situations: 1. Low threshold: Large mismatch accepted; Few large clusters; 2. High threshold: Small mismatch accepted; Many small clusters; Higher precision. Def and Function: A user-chosen design parameter to control the sensitivity of the clustering value larger, data are homogenous in cluster.

Associative Memory: Two most important cognitive functions in brain-like intelligence: Learning and memory; Different from the linear address storage mechanism of modern computers; When given a probe, the associate memory should converge to an equilibrium in a robust and fault-tolerant way.

Mechanism: Content-addressable mechanisms; Store prototype patterns which can be retrieved with the recalling cues.

Auto-association vs. Hetero-association (Subtype): Auto-associative Memory: retrieves a previously stored pattern that closely resembles the recalling probe.

Hetero-associative Memory: Input and output are not the same thing, but there is a logical correspondence between them.

Memory Processes: Storage(Information encoding): Given a set of prototype patterns to be memorized, place them into the memory indefinitely. Retrieval (Information decoding): Given any probe (key or cue), recall the corresponding prototype patterns in the memory.

Bidirectional Associative Memories: 1. Known as hetero-associative memories and resonance networks; 2. A generalization of auto-associative memories; 3. Use bipolar signum activation functions.

Hopfield Networks(application of associate memory): Application: associative memories; optimization models. Structure: Single-layer recurrent neural networks. Two variants: Discrete-time model uses bipolar threshold logic units; Continuous-time model uses unipolar sigmoid activation function.

Stability Conditions: Def:

Sufficient conditions: 1. Symmetrical connection & no self-connection: 2. Asynchronous Updating-only one neuron is updated at a time.

Stability Properties: Weight matrix W is symmetric with zero diagonal elements+the activation is conducted asynchronouslythe discrete-time Hopfield network is stable (Sufficient condition)

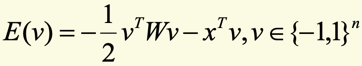
Weight matrix W is symmetric with zero diagonal elements+the activation is conducted synchronously Network is either stable or oscillates in a limit cycle of two states

Limitations (Hopfield Network): 1. Very

limited capacity: , where n is the memory

length. 2. Many spurious states; e.g., .

Discrete-Time Hopfield Network as an Optimization Model: 1. Formulate the energy function according to the objective function

and constraints.  2. Form

a Hopfield network, then update the states asynchronously until convergence. Shortcoming: slow convergence due to asynchrony.

Continuous-Time Hopfield Network as an Optimization Model: 1. Formulate the energy function according to the objective function

and constraints; 2. An

equilibrium state is a local minimum of the energy function.

Traveling Salesman Problem (continuous Hopfield network): Checking out all possible routes: (N - 1)! / 2; A continuous Hopfield network can compute a good solution in a parallel and distributed manner; Parameter Selection: *A* = *B* =*D* =250, *C* =1000 and ξ= 50, The optimal path can be found in approximately 50% of cases.

Simulated Annealing: Def: simulates the physical annealing process mathematically; Used in global optimization of nonconvex objective function.

Characteristics of Simulated Annealing: 1. High temperatures are easy to accept differential solutions; 2. As the temperature approaches to zero, the procedure becomes an iterative improvement one; 3. Temperature parameter has to be lowered gradually to avoid prematurity.

Boltzmann Machine: A stochastic recurrent neural network; Binary state variables {-1, 1}*n* with a probabilistic activation function; A parallel implementation of simulated annealing procedure; A stochastic, generative counterpart of the Hopfield networks.

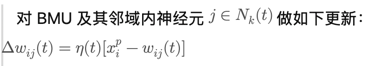
Mean Field Annealing Network: A deterministic recurrent neural network; Based on mean-field theory; Continuous state variables on [-1, 1]*n*; Bipolar sigmoid activation function; Use a gradual decreasing temperature parameter like simulated annealing; For combinatorial optimization.

Self-Organizing Maps: Single-layer, winner-take-all, unsupervised learning; Formation of topographic map through self-organization; Map high-dimensional data to 1D or 2D feature maps.

Kohonen’s Learning Algorithm: 1. Initialization: Randomize wij(0). 2. Compute

distance for datum xp.  3.

Minimization: Find k such thatdk = minjdj. 4.

Adaptation: 

Cycle control part: 0<<1: Learning rate control amplitude; < 0, Learning rate decreases over time; p p+1: Change to the next input data point; Return to Distance and continue iterating until it converges.

Echo State Network: RNN with sparse connections and random weights among hidden neurons.

Fuzzy Logic: A generalization of classical logic; Fuzzy logic → impreciseness / ambiguity; Probability → randomness.

Membership Function: uA: X-> [0, 1].

Crisp set: Only 0 or 1. Fuzzy set: Allow "between".

Fuzzy Set A: the set of all pairs (*x*, *u*A(*x*))

where *x* belongs to *X;* i.e*.,* .

Properties of Fuzzy Relations: Reflexive, Symmetric, Transitive.

Linguistic Variables: Def: values are words or sentences in natural or artificial languages. Important in fuzzy logic and approximate reasoning. Example: speed can be defined as a linguistic variable and takes values of slow, fast, and very fast.

Fuzzifiers: A mapping from a real-valued set to a fuzzy by means of a membership function.

Defuzzifiers: A mapping from a fuzzy set to a real-valued set. Two methods for Defuzzifiers: Centoid defuzzifier, Center average defuzzifier.

Fuzzy Inference Process: Fuzzification Inference Defuzzification(centoid method)

Type-2 Fuzzy Logic: A generalization of type-1 fuzzy logic to handle the uncertainty of membership function by using fuzzy membership function.

Evolutionary Computation: Population-based stochastic and meta-heuristic search algorithms for global or multi-objective optimization. Use collective wisdom to accomplish given tasks via efforts of many generations.

Genetic Algorithms (Belongs to EC): Def: A stochastic search method simulating the evolution of population of living species; Optimize a fitness function which is not necessarily continuous or differentiable; A genetic algorithm generates a population of seeds instead of one in traditional algorithms; The computation of the population can be carried out in parallel.

Elements in Genetic Algorithms: Coding: produce the required discretization of decision variables in terms of strings. Reproduction / Selection: copy individual strings according to their fitness. A set of information-exchange operators-crossover, generate new and better population of points. Mutation: modify data, inject new information into offspring (Reproduction and crossover produce new string without introducing new information).

Swarm Intelligence: made up of a population of simple agents interacting locally with one another and with their environment.

Typical representatives: particle swarm optimization, ant colony optimization.

Particle swarm optimization: A robust stochastic optimization technique based on the movement and intelligence of swarms; Applies the concept of social interaction to problem solving; Use a number of agents that constitute a swarm moving around in the search space looking for the best solution; Each particle is treated as a point in a multi-dimensional space.

Perceptron Learning Algorithm. goal: find appropriate weight w and bias b so data points being correctly classified. Algorithm: 1. Initialize parameters (weight, bias and learning rate); 2. For each sample i in the training set D, perform the following steps: a. Compute the predicted value: , b. If mis classified update the weights: update the bias: . Repeat step 2 until all data points are correctly classified or the maximum number of iterations is reached.

MLPs are neural network models that work as universal approximators, i.e., MLPs are able to approximate any continuous function, rather than only linear functions.

a XOR b = (a AND NOT b) OR (b AND NOT a)

MLP: Example in XOR function. Why These Specific Weights? These specific weights and biases are hand-designed to accurately simulate the logical behavior of the XOR.

h1 = σ((1 - x1) + x2) = σ((-1)x1 + x2 + 1)

h2 = σ(x1 + (1 - x2)) = σ(x1 + (-1)x2 + 1)

y = σ(h1 + h2) = σ(h1 + h2 + 0)

Why single perceptron is not enough? It can’t solve the non-linear problem like XOR gate; Three-variable parity problem.

Why are the linear activation functions and hard-limiter activation functions not useful in the hidden layers of a MLP with the back-propagation learning algorithm? 1.Linear activation functions keep the network as a linear transformation, making it equivalent to a single-layer perceptron, unable to learn nonlinear problems like XOR; 2.The hard-limiter (step function) has zero derivative almost everywhere, so gradients vanish and backpropagation cannot update weights, making learning impossible.

RBF network classifies linearly inseparable data, like the XOR problem.

**Feedforward network & recurrent network:** Feedforward networks are suitable for static input-output mapping, MLP; Feedback networks are suitable for dynamic problems that require the memory of historical states, Hopfield Network.

**Single-layer vs. Multi-layer Models**: Multi-layer models fit training data better and may generalize better if enough training data is available. Single-layer models have better relative generalization and are faster to train and execute. Use single-layer when speed is critical or the improvement from multi-layer is small. Choose multi-layer if it significantly improves fit and enough training examples are provided.

**Deterministic & stochastic**: The output of the Deterministic model is fixed, while the output of the Stochastic model is random. The former is applicable to the determination system, while the latter is closer to the real world.

The difference between **RBF and MLP**: RBF uses localized activation, learns in two steps, and suits structured approximation; MLP uses global activation, learns via backpropagation, and is more general-purpose.

**Perceptron** vs. **Adaline**: Architecture: Perceptron, bipolar or unipolar hardlimiter activation function; Adaline, linear activation function; Learning rule: perceptron learning algorithm, not gradient-descent and in sequential or batch training mode; Adaline learning algorithm, gradient descent and only operate in batch mode.

**Number of Logic Functions** vs. **Number of Threshold Functions**: threshold, ; logic, .

**Swarm Intelligence** and **Evolutionary Computation** are two different natural heuristic optimization methods and both belong to the category of **computational intelligence**.

**Supervised Learning Models in FNN:** FNN: One-way signal flow; no feedback or memory. Used in classification or regression (e.g., image inputs, not just labels). Supervised Learning: All models listed are trained with input-output pairs. Including: TLU, Perceptron, Adaline, MLP, RBF, SVM, FLN, ELM.

**RNN:** Designed for problems where output is not known in advance; Involves longer computation, often optimization-based; Has internal memory – must “think” (iterate) before giving a final answer; Often used in unsupervised or dynamic pattern recognition tasks. Including: TLU, deep learning, Hopfield network, ART, SOM.

**MLP**: Pros: have universal approximations. Cons: Local minima problem; Generalization problem; no details for finding solutions.

**Deep learning**: Pros: capture massive details; sophisticated architecture; Low generalization error. Cons: Requires powerful hardware; Slower convergence due to large number of weights & vanishing gradient; Lacks theoretical guidance.

**MLP vs Deep Learning**: Similarities: Both are multilayer neural networks (MLP ⊂ DNN); Both can be feedforward networks; No clear theory on required number of layers/neurons. Difference: Generalization: MLP tends to have generalization problems; Deep learning achieves low generalization error with sufficient data; Network Structure: MLP is strictly feedforward, no cycles; Deep learning can be feedforward, recurrent, or cyclic; Layer Types: MLP uses only fully connected layers with simple activations (e.g., sigmoid, tanh); Deep learning supports CNN, RNN, FC, and other layer types.

**RBF:** Pros: Avoids local minima via reachability-based output layer; Fast & accurate when using randomly fixed hidden neurons; Theoretically and experimentally validated. Cons: Transferability issue: Hard to determine RBF centers (e.g., ci); Random hidden neurons may be suboptimal (like coin toss); Only output weights are trainable; hidden layer is fixed.

**ELM:** Pros: Can avoid local minima problem; incremental ELM is a universal approximator. Cons: Similar to RBF.

**FLN:** Pros: Acts as a universal approximator; One-hidden-layer network with heterogeneous, randomly generated nodes; Fast and accurate (e.g., RVFL). Cons: Like RBF, the hidden layer is fixed and not trainable; Only output weights are updated during training; Performance depends on random initialization.

**FLN vs ELM**: Similarity: Both randomly generate hidden neurons; Only output weights are trained, hidden layer is fixed. Difference: FLN allows direct input-to-output connections; ELM does not allow direct input-output connections.

**Type-1 Fuzzy Set**: Pros: Effective in manypractical applications. Cons: Limited ability to model uncertainty in real-world data; Represents uncertainty using a single membership function (MF) only; Lacks a mechanism to express dispersion in linguistic terms.

**Type-2 Fuzzy Set**: Pros: Captures uncertainty in membership functions; Provides a measure of dispersion, similar to how variance represents spread in statistic; More effective than Type-1 FS for handling linguistic and data uncertainties.

**Type-1 vs Type-2 Fuzzy Sets**: Similarities: Both handle linguistic and random uncertainty; Both are used in fuzzy logic systems (FLSs) to deal with imprecise or noisy information; If the membership function (MF) itself is fuzzy, it’s a Type-2 FS. Differences:Uncertainty Handling: T1 FS uses crisp MFs with limited ability to model uncertainty; T2 FS models uncertainty in the MF itself, better for vague or noisy data; Membership Function: T1 FS is crisp; T2 FS is fuzzy-fuzzy; System Usage: T1 FLS uses only T1 fuzzy sets; T2 FLS includes at least one T2 fuzzy set.

**SVM:** Pros: Efficient computation in high-dimensional spaces; Strong generalization ability; Supported by solid theoretical foundations.

**LSSVM:** Reformulates SVM as a set of linear equations; Solves a convex optimization problem (no local minima).Pros: Low computational cost; Excellent generalization, even on complex tasks (e.g., two-spiral classification); Performs well over a wide range of kernel (σ) and regularization (γ) parameters.

**SVM vs LSSVM**: Similarities: Both maximize the margin between classes (not minimize error); Both apply Mercer’s condition; Both handle binary classification with support vectors. Differences:Optimization Formulation: SVM solves a Quadratic Programming Problem (QPP); LSSVM solves a linear set of equations, which is simpler; Support Values: In SVM, only near-boundary points have non-zero support values; In LSSVM, support values are proportional to errors, so all points can contribute; Constraint Type: SVM is based on inequality constraints; LSSVM uses equality constraints, making it easier to solve; Performance & Complexity: SVM is not especially fast or efficient in general cases; LSSVM offers low computational cost with strong generalization (e.g., two-spiral problem).

**Genetic Algorithm:** Pros: No derivatives required; Highly parallelizable; Escapes local minima through diverse population; Good global search capability. Cons: No guarantee of convergence; Slow convergence, especially in fine-tuning; Requires many function evaluations; Parameter tuning is heuristic.

**Ant Colony Optimization:** Adv: Inherent parallelism; Positive Feedback accounts for rapid discovery of good solutions; Efficient for Traveling Salesman Problem and similar problems. Dev:  Theoretical analysis is difficult; Sequences of random decisions (not independent); Probability distribution changes by iteration.

**Particle Swarm Optimization:** Pros: Simple computation, easy to implement; Fast convergence via information sharing from the best particle; No crossover or mutation operations needed. Cons: May suffer from premature convergence (partial optimism); Not effective for scattered or highly irregular search spaces; Struggles with non-coordinate systems.

**GA vs ACO:** Similarities: Both are suitable for TSP and other combinatorial problems; Both are prone to premature convergence (local optima); Parameter tuning is experimental, not theoretically guaranteed; No special assumptions on the search space (e.g., derivatives, continuity, convexity).

Difference: GA searches faster but lacks feedback, which may lead to redundant or inefficient iterations; ACO uses pheromone feedback to guide the search but converges more slowly at the beginning; For larger problems, ACO generally performs better than GA, though it may also stagnate when the problem size is too big.

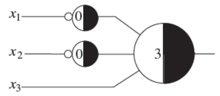
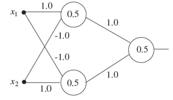
**GA vs PSO:** Similarities: Both are nature-inspired optimization algorithms; Widely used and effective for complex optimization problems. Differences: PSO typically converges faster (fewer generations) than GA; GA has lower per-generation computation time, while PSO may consume more time per generation despite faster convergence overall.

**ACO vs PSO**: Similarities: Both are population-based stochastic optimizationalgorithms; Inspired by social behavior (ants for ACO, particles for PSO), unlike GA which mimics biological evolution; Effective in global optimization, especially for combinatorial and mixed-integer problems; Less prone to local optima than gradient-based methods. Differences: PSO is typically faster and easier to implement, with fewer parameters to tune; ACO is generally more robust and accurate, especially on complex or constraint-based problems.

**SVR**: ✅ Extension of SVM for regression problems; ✅ Introduces ε-insensitive loss to ignore small errors; ❌ Similar computational burden as SVM; ❌ Requires careful tuning of ε, C, and kernel parameters.

**Activation Functions:** **Binary Step Function**: f(x) = 1 if x ≥ 0; else 0. ✅ Simple, good for binary decisions; ❌ Gradient is always 0 → no learning via backpropagation; ❌ Not suitable for multi-class tasks; ❌ Theoretical only, rarely used in practice. **Linear Function**: f(x) = ax. ✅ Easy to interpret; used in output layers for regression; ❌ Gradient is constant → no deep learning effect; ❌ Output is always linear → network collapses to a linear model. **Sigmoid Function**: f(x) = 1 / (1 + e^-x). ✅ Non-linear, output between 0 and 1; ✅ Smooth, differentiable → supports backpropagation; ❌ Vanishing gradient problem outside [-3, 3]; ❌ Output always positive → not zero-centered. **Tanh Function**: f(x) = tanh(x) = 2 / (1 + e^(-2x)) – 1. ✅ Non-linear, differentiable, output between -1 and 1; ✅ Zero-centered, preferred over sigmoid; ❌ Still suffers from vanishing gradient issue. **ReLU**: f(x) = max(0, x). ✅ Non-linear, simple, fast, sparse activation; ✅ Efficient for deep networks, most widely used; ❌ Dying ReLU problem: zero gradient for x < 0 → dead neurons.

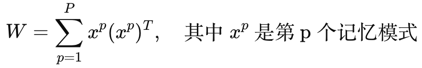
Decoder for (0, 0, 1): 下左

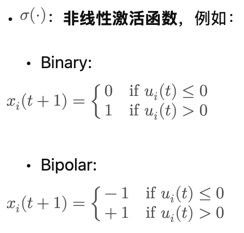
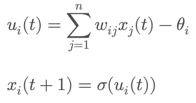
XOR Problem Solved: 上右。

Discrete Hopfield Network, DHN:

权重通过 Hebbian 学习规则得到:

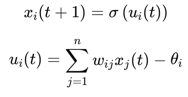
****

更新规则 Updating Rule: 在时间t，对每个神经元 ii，进行如下更新:

****

稳定性 Stability: 每次更新都会使能量下降，直到局部最小值，即稳定状态。

离散网络的平衡判定条件(equilibrium):

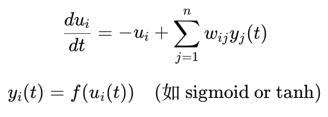
** **

x(t)是一个平衡态.

能量函数:

****

连续网络的平衡判定条件(equilibrium):

****

能量函数:

** **

**Discrete Hopfield Network VS Continuous Hopfield Network**: DHN: State: Binary (0/1) or Bipolar (−1/+1); Time: Discrete updates; Activation: Step function; Easy to implement and simulate; Converges to stable states via energy minimization; Best for pattern storage and associative memory. CHN: State: Continuous real values; Time: Continuous dynamics (ODE-based); Activation: Smooth functions (e.g., sigmoid, tanh); Requires solving differential equations; Also minimizes energy under symmetric weights; Suitable for optimization and analog signal tasks.