Self-Organizing Maps (SOMs)

Unsupervised Neural Networks for Data Visualization

Kiran Bagale

SXC

August, ©025

7.1 Self-Organizing Map

- **Self-Organizing Map (SOM)** or **Self-Organizing Feature Map (SOFM)**
- Unsupervised machine learning technique
- Produces low-dimensional representation of high-dimensional data
- Preserves topological structure of the data
- Visualizes clusters of observations with similar values

Key Feature

High-dimensional data $\rightarrow$ 2D "map" where proximal clusters have similar values

Historical Background

- Introduced by Finnish professor **Teuvo Kohonen** in the 1980s
- Also known as **Kohonen Map** or **Kohonen Network**
- Built on:
  - Biological models of neural systems (1970s)
  - Morphogenesis models by Alan Turing (1950s)
- Creates representations similar to cortical homunculus

Inspiration

Based on how sensory information is processed in separate parts of the cerebral cortex

Feature-Mapping Models: Neurobiological Inspiration

- The cerebral cortex shows remarkable **orderly mapping** of sensory inputs
- Different sensory inputs are mapped onto corresponding cortical areas
- **Principle of topographic map formation:**

Key Principle

"The spatial location of an output neuron in a topographic map corresponds to a particular domain or feature of data drawn from the input space."

- Four key properties of computational maps:
  1. Neurons process similar information in parallel
  2. Information context is preserved at each stage
  3. Related neurons are spatially close together
  4. Maps represent decision-reducing mappings from high-dimensional spaces

Two Self-Organized Feature Maps

Willshaw–von der Malsburg Model (1976)

- Biological motivation
- Retinotopic mapping problem
- Two interconnected 2D lattices
- Short-range excitatory mechanism
- Long-range inhibitory mechanism
- Same input/output dimensions

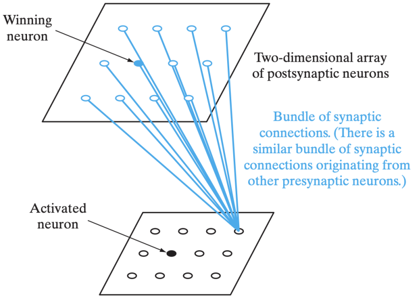
Kohonen Model (1982)

- Computational focus
- Vector quantization approach
- Data compression capability
- More general applicability
- Dimensionality reduction
- Encoder-decoder structure

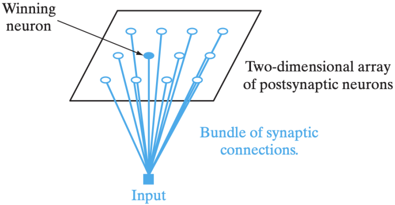
Common Features:

- Two-dimensional lattice of output neurons
- Self-organization through learning
- Topologically ordered mappings

Two self-organized feature maps



(a) Willshaw–von der Malsburg's model



(b) Kohonen model

Model Characteristics and Comparison

Willshaw–von der Malsburg Model

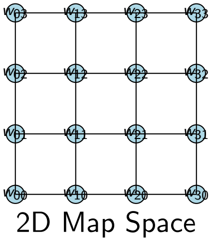
- **Mechanism:** Hebbian-type modifiable synapses
- **Specialization:** Retinotopic mapping in visual cortex
- **Limitation:** Input dimension = Output dimension
- **Key feature:** Geometric proximity coding through electrical activity correlations

Kohonen Model

- **Class:** Vector-coding algorithm
- **Capability:** Data compression and dimensionality reduction
- **Approach:** Places fixed number of code vectors in high-dimensional input space
- **Advantage:** More computationally tractable and general
- **Applications:** Traditional approach in neural networks (Kohonen, 1982, 1990, 1997)

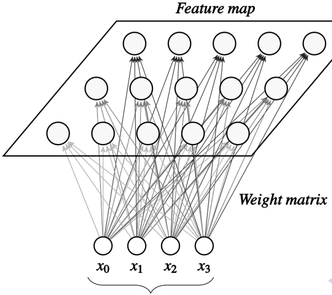
Two main components:

1. **Input Space**: p-dimensional data
2. **Map Space**: 2D grid of nodes/neurons



Node arrangement:

- Hexagonal or rectangular grid
- Number of nodes specified beforehand
- Each node has a weight vector



SOM Algorithm - Core Components

- **Input Space:** Continuous activation patterns with probability distribution
- **Network Topology:** Lattice of neurons defining discrete output space
- **Neighborhood Function:** $h_{j,i(x)}(n)$ around winning neuron $i(x)$
- **Learning Rate:** $\eta(n)$ starting at η_0 and decreasing with time n

Key Equations:

$$h_{j,i(x)}(n) = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2(n)}\right) \tag{1}$$

$$\eta(n) = \eta_0 \exp\left(-\frac{n}{\tau_2}\right) \tag{2}$$

SOM Algorithm Steps

1. **Initialization:** Choose random weight vectors $\mathbf{w}_j(0)$
   - Keep weights small and different for all neurons
2. **Sampling:** Draw sample $\mathbf{x}$ from input space
3. **Similarity Matching:** Find winning neuron using minimum distance

$$i(\mathbf{x}) = \arg\min_j \|\mathbf{x}(n) - \mathbf{w}_j\| \tag{3}$$

4. **Updating:** Adjust weights of excited neurons

$$\mathbf{w}_j(n+1) = \mathbf{w}_j(n) + \eta(n) h_{j,i(x)}(n)(\mathbf{x}(n) - \mathbf{w}_j(n)) \tag{4}$$

5. **Continuation:** Repeat until convergence

Operating Modes

1. Training Mode

- Uses input data set to generate lower-dimensional representation
- Moves weight vectors toward input data
- Preserves topology from map space

2. Mapping Mode

- Classifies new input data using generated map
- Finds node with closest weight vector
- Uses distance metrics (e.g., Euclidean distance)

Two Phases of Learning

1. Ordering (Global Organization) Phase

- Duration: $\sim$1,000 iterations
- Large neighborhood function
- Topological ordering occurs
- Parameters:

$$\eta_0 = 0.1 \tag{5}$$

$$\tau_1 = \frac{1000}{\log \sigma_0} \tag{6}$$

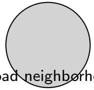
2. Convergence (Fine Tuning) Phase

- Duration: $500\times$ # neurons
- Small neighborhood ($\sim$0.01)
- Fine-tuning of feature map
- Statistical quantification
- Prevents metastable states

Late Training

Narrow neighborhood

Early Training



Broad neighborhood

Critical: Learning rate $\eta(n)$ must never reach zero to avoid network getting stuck in metastable states.

Neighborhood Function

Gaussian Neighborhood Function:

$$h_{j,i(x)} = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2}\right) \tag{7}$$

Properties:

- Symmetric around winning neuron
- Decreases monotonically with distance
- Width σ shrinks over time
- Translation invariant

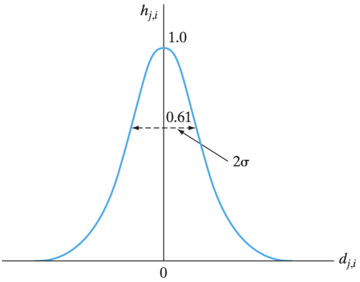


Figure: Gaussian neighborhood function

Time-varying width:

$$\sigma(n) = \sigma_0 \exp\left(-\frac{n}{\tau_1}\right) \tag{8}$$

Three Essential Processes

1. **Competition**
   - Neurons compute discriminant function values
   - Winner-takes-all mechanism
   - Neuron with largest inner product wins
2. **Cooperation**
   - Winning neuron determines spatial location of topological neighborhood
   - Neighboring neurons cooperate in learning
   - Based on lateral distance in output space
3. **Synaptic Adaptation**
   - Excited neurons adjust synaptic weights
   - Move weight vectors toward input pattern
   - Enhanced response to similar future inputs

Adaptive Process - Weight Updates

Modified Hebbian Learning with Forgetting:

$$\Delta \mathbf{w}_j = \eta y_j \mathbf{x} - g(y_j)\mathbf{w}_j \tag{9}$$

For winning neuron: $y_j = h_{j,i(x)}$, leading to:

$$\Delta \mathbf{w}_j = \eta h_{j,i(x)}(\mathbf{x} - \mathbf{w}_j) \tag{10}$$

Key Features:

- Forgetting term prevents weight saturation
- Topological ordering emerges naturally
- Adjacent neurons develop similar weight vectors
- Creates organized feature map

SOM Summary and Applications

Key Advantages:

- Dimensionality reduction while preserving topology
- Unsupervised learning capability
- Robust to noise and variations
- Visualizable output space

Applications:

- Data visualization and clustering
- Pattern recognition
- Feature extraction
- Sensory mapping models

Core Principle: A continuous input space of activation patterns is mapped onto a discrete output space of neurons through competitive learning with topological cooperation.

7.3.Properties of the Feature Map

- Once converged, the SOM algorithm displays important statistical characteristics of the input space
- The feature map Φ is a nonlinear transformation: $\Phi : \mathcal{X} \rightarrow \mathcal{A}$
- Maps continuous input space $\mathcal{X}$ onto discrete output (lattice) space $\mathcal{A}$
- SOM embodies two key ingredients:
    1. **Projection**: from continuous input data space to discrete neural space
    2. **Pointer**: from output space back to input space via weight vectors

1. Vector Quantization Capability

Key Property

The feature map Φ, represented by the set of synaptic weight vectors $\{w_i\}$ in the output space $\mathcal{A}$, provides a good approximation to the input space $\mathcal{X}$.

- SOM is fundamentally a **vector quantization algorithm**
- Provides efficient dimensionality reduction and data compression
- Weight vectors act as prototypes representing clusters in input space
- Theoretical basis rooted in vector quantization theory (Gersho and Gray, 1992)

2. Spatial Organization Preservation

Topological Ordering Property

The feature map Φ computed by the SOM algorithm is topologically ordered in the sense that the spatial location of a neuron in the lattice corresponds to a particular domain or feature of input patterns.

- Direct consequence of the update equation forcing winning neuron weights toward input vector
- Creates an **elastic or virtual net** with lattice topology
- Feature map displayed in input space shows neighboring relationships
- Preserves neighborhood structure from input to output space

3. Statistical Representation

Density Matching Property

The feature map Φ reflects variations in the statistics of the input distribution: Regions in the input space $\mathcal{X}$ from which sample vectors x are drawn with a high probability of occurrence are mapped onto larger domains of the output space.

- Magnification factor $m(x)$ relates to input probability density $p_X(x)$
- Ideal relationship: $m(x) \propto p_X(x)$
- Two encoding methods yield different relationships:
  1. **Minimum-distortion encoding**: $m(x) \propto p_X^{2/3}(x)$
  2. **Nearest-neighbor encoding**: $m(x) \propto p_X^{2/3}(x)$

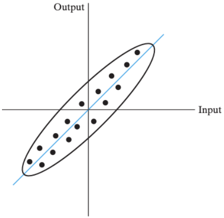
4. Automatic Feature Extraction

Feature Selection Property

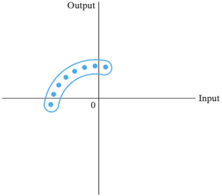
Given data from an input space, the self-organizing map is able to select a set of best features for approximating the underlying distribution.

- Natural culmination of Properties 1-3
- Similar to principal components analysis but with important differences
- Handles both **linear** and **nonlinear** input-output mappings
- Figure below illustrates:
  - (a) Linear distribution $\rightarrow$ linear mapping
  - (b) Nonlinear distribution $\rightarrow$ nonlinear mapping adaptation

Key Implementation Notes



(a)



(b)

- **Batch SOM**: Rewrite using summations to approximate integrals
- No learning-rate schedule required in batch version
- Still requires neighborhood function for topological ordering
- Neighborhood function $h_{j,i(x)}$ has form of probability density function
- Zero-mean Gaussian model appropriate for noise modeling
- **Important limitation**: SOM tends to overrepresent regions of low input density and underrepresent high-density regions

Connection to Vector Quantization Theory

Generalized Lloyd Algorithm

SOM can be viewed as batch training version of generalized Lloyd algorithm with conditions:

- **Condition 1**: Choose code $c = c(x)(Encoder)$ to minimize squared-error distortion
- **Condition 2**: Compute reconstruction vector as centroid of input vectors satisfying Condition 1
- Expected distortion: $D = \frac{1}{2} \int_{-\infty}^{\infty} p_X(x) ||x - x'||^2 dx$
- Operates in batch training mode with alternating optimization

7.4 Contextual Maps

- In contextual mapping, neurons in 2D lattice are partitioned into **coherent regions**
- Each region represents distinct sets of contiguous symbols or labels
- Process:
  1. Train SOM on input data
  2. Present test patterns to trained network
  3. Assign labels to neurons based on strongest responses
  4. Group similar responses into semantic clusters
- Results in maps resembling cortical organization (e.g., visual cortex)
- Two main visualization methods:
  1. **Elastic net**: Feature map viewed as elastic net with synaptic weights as pointers
  2. **Contextual maps**: Class labels assigned to neurons based on test pattern responses
- Applications include data mining, pattern recognition, and exploratory data analysis

Example: Animal Classification

- Dataset: 16 animals with 13 binary attributes
- Attributes include: size, legs, habitat preferences, etc.
- Input vector construction:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_u \end{bmatrix} = \begin{bmatrix} \mathbf{x}_a \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{x}_u \end{bmatrix}$$

- Parameter $a = 0.2$ balances attribute vs. symbol code influence

Semantic clusters found:

- Birds (white region)
- Peaceful species (grey)
- Hunters (blue)

Animal Classification

Animal		Dove	Hen	Duck	Goose	Owl	Hawk	Eagle	Fox	Dog	Wolf	Cat	Tiger	Lion	Horse	Zebra	Cow
is	small	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
	medium	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	big	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
has	2 legs	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	4 legs	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	hair	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	hooves	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	mane	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
	feathers	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
likes to	hunt	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
	run	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
	fly	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
	swim	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Figure: Animal Names and Their Attributes

Animal Classification

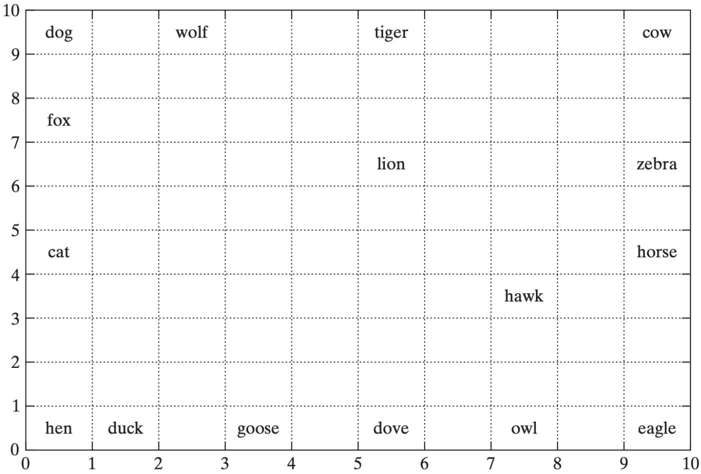


Figure: Feature map containing labeled neurons with strongest responses to their respective inputs

Animal Classification

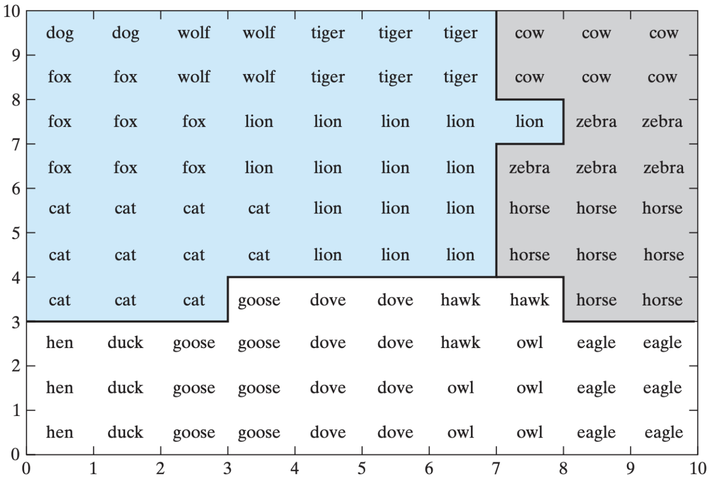


Figure: Semantic map obtained through the use of simulated electrode penetration mapping. The map is divided into three regions, representing birds (white), peaceful species (grey), and hunters (blue).

7.5 Hierarchical Vector Quantization

- Vector quantization is a form of **lossy data compression**
- Based on rate distortion theory from Shannon's information theory
- Fundamental result (Gray, 1984):

Rate Distortion Theory

Better data compression performance can always be achieved by coding vectors instead of scalars, even if the source of data is memoryless or if the data compression system has memory.

- Conventional vector quantization requires prohibitive computation
- Most time-consuming part: encoding operation
- For N code vectors, encoding time is $O(N)$

Multistage Hierarchical Vector Quantizer

- Solution: Trade off accuracy for speed of encoding
- Factor overall vector quantization into suboperations
- Each suboperation requires very little computation
- Uses table lookup per suboperation

Two-Stage Process

Consider two vector quantizers VQ_1 and VQ_2:

- VQ_1 feeds its output into VQ_2
- Output from VQ_2 is the final encoded version
- VQ_2 must account for distortion induced by VQ_1

- Training method: SOM algorithm for all stages except the last
- Last stage: generalized Lloyd algorithm

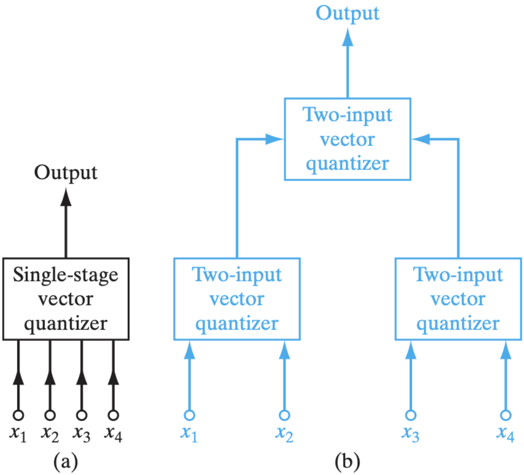


Figure: (a) Single-stage vector quantizer with four- dimensional input. (b) Two-stage hierarchical vector quantizer using two-input vector quantizers.

Limitations of Standard SOM

Kohonen's SOM has two fundamental limitations:

1. **Accuracy Issue:** The estimate of the probability density function of the input space lacks accuracy
   - Shows up in experimental results
   - Density-matching property is imperfect
2. **No Objective Function:** The algorithm has no objective function that could be optimized
   - Makes convergence proofs difficult
   - Nonlinear stochastic characterization

Solution: Kernel-based formulation developed by Van Hulle (2002b)

7.6 Kernel SOM: Objective Function

- In kernel SOM, each neuron acts as a kernel
- Kernel parameters are adjusted individually
- Uses a prescribed **objective function**
- Focus: joint entropy of kernel outputs

Differential Entropy

For continuous random variable Y_i with probability density $p_Y(y_i)$:

$$H(Y_i) = -\int_{-\infty}^{\infty} p_Y(y_i) \log p_Y(y_i) dy_i$$

Bottom-up Training Process

1. Maximize differential entropy of each kernel
2. Adjust kernel parameters to maximize mutual information between kernel output and input

Kernel Definition

Kernel Notation

Kernel: $k(\mathbf{x}, \mathbf{w}_i, \sigma_i)$

- $\mathbf{x}$: input vector (dimensionality m)
- $\mathbf{w}_i$: weight vector of i-th kernel
- σ_i: width of i-th kernel
- $i = 1, 2, \ldots, l$ (total number of neurons)

Radially Symmetric Kernel

$$k(\mathbf{x}, \mathbf{w}_i, \sigma_i) = k(\|\mathbf{x} - \mathbf{w}_i\|, \sigma_i), \quad i = 1, 2, \ldots, l$$

where $\|\mathbf{x} - \mathbf{w}_i\|$ is the Euclidean distance.

- Uses Gaussian probability distribution for kernel definition
- Kernel output has "bounded" support for maximum differential entropy

Statistical Foundation

Input Vector Assumptions

- m elements of input vector $\mathbf{x}$ are statistically independent and identically distributed (iid)
- j-th element is Gaussian distributed: mean μ_j, variance σ^2
- Mean vector: $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_m]^T$

Chi-square Distribution

Squared Euclidean distance: $v = \|\mathbf{x} - \boldsymbol{\mu}\|^2 = \sum_{j=1}^{m}(x_j - \mu_j)^2$
Random variable V has chi-square distribution:

$$p_V(v) = \frac{1}{\sigma^m 2^{m/2}\Gamma(m/2)} v^{(m/2)-1} \exp\left(-\frac{v}{2\sigma^2}\right), \quad v \geq 0$$

Radial Distance Distribution

Transformation to Radial Distance

Radial distance: $r = v^{1/2} = \|\mathbf{x} - \boldsymbol{\mu}\|$
Probability density function:

$$p_R(r) = \begin{cases} \frac{1}{2^{(m/2)-1}\Gamma(m/2)} \left(\frac{r}{\sigma}\right)^{m-1} \exp\left(-\frac{r^2}{2\sigma^2}\right), & r \geq 0 \\ 0, & r < 0 \end{cases}$$

Large Dimension Approximation

For large m:

$$\mathbb{E}[R] \approx \sqrt{m}\sigma, \quad \mathrm{Var}[R] \approx \frac{\sigma^2}{2}$$

Final Kernel Definition

Incomplete Gamma Distribution

Cumulative distribution function:

$$P_R(r|m) = 1 - \frac{\Gamma\left(\frac{m}{2}, \frac{r^2}{2\sigma^2}\right)}{\Gamma\left(\frac{m}{2}\right)}$$

The complement of incomplete gamma distribution provides the desired kernel.

Kernel SOM Formula

$$k(\mathbf{x}, \mathbf{w}_i, \sigma_i) = \frac{1}{\Gamma\left(\frac{m}{2}\right)}\Gamma\left(\frac{m}{2}, \frac{\|\mathbf{x} - \mathbf{w}_i\|^2}{2\sigma_i^2}\right), \quad i = 1, 2, \ldots, l$$

- As input-space dimensionality m increases, $p_R(r)$ approaches Gaussian function
- Provides improved topographic mapping compared to standard SOM

7.7 Kullback-Leibler Divergence (KLD)

Definition

KLD provides a formula for assessing the quality of a density estimate against the true density.

For true density $p_X(\mathbf{x})$ and estimate $\hat{p}_X(\mathbf{x})$:

$$D_{p_X \| \hat{p}_X} = \int_{-\infty}^{\infty} p_X(\mathbf{x}) \log\left(\frac{p_X(\mathbf{x})}{\hat{p}_X(\mathbf{x})}\right) d\mathbf{x}$$

Properties

- KLD is always a nonnegative number
- KLD $= 0$ if and only if $\hat{p}_X(\mathbf{x})$ matches $p_X(\mathbf{x})$ exactly
- Used in information theory terminology

Gaussian Mixture Density Estimate

Mixture Model

Density estimate expressed as mixture of Gaussian density functions:

$$\hat{p}_X(\mathbf{x}|\mathbf{w}_i, \sigma_i) = \frac{1}{l}\sum_{i=1}^{l} \frac{1}{(2\pi)^{m/2}\sigma_i^m} \exp\left(-\frac{1}{2\sigma_i^2}\|\mathbf{x} - \mathbf{w}_i\|^2\right)$$

- Conditional on weight vector $\mathbf{w}_i$ and width σ_i for $i = 1, 2, \ldots, l$
- Equal mixing coefficients $(1/l)$
- Optimal density estimate $p_X(\mathbf{x})$ obtained by minimizing KLD

Goal

Find optimal parameters $\mathbf{w}_i$ and σ_i that minimize KLD between true and estimated densities.

KLD Optimization: Partial Derivatives

Partial Derivative w.r.t. Weight Vector

$$\frac{\partial}{\partial \mathbf{w}_i}(D_{p_X \| \hat{p}_X}) = -\int_{-\infty}^{\infty} p_X(\mathbf{x}) \left(\frac{1}{\hat{p}_X(\mathbf{x}|\mathbf{w}_i, \sigma_i)} \frac{\partial}{\partial \mathbf{w}_i} \hat{p}_X(\mathbf{x}|\mathbf{w}_i, \sigma_i) \right) d\mathbf{x}$$

Partial Derivative w.r.t. Width

$$\frac{\partial}{\partial \sigma_i}(D_{p_X \| \hat{p}_X}) = -\int_{-\infty}^{\infty} p_X(\mathbf{x}) \left(\frac{1}{\hat{p}_X(\mathbf{x}|\mathbf{w}_i, \sigma_i)} \frac{\partial}{\partial \sigma_i} \hat{p}_X(\mathbf{x}|\mathbf{w}_i, \sigma_i) \right) d\mathbf{x}$$

- Setting partial derivatives to zero gives optimization conditions
- Use stochastic approximation theory (Robbins and Monro, 1951)

Learning Rules from KLD Minimization

Weight Update Rule

$$\Delta \mathbf{w}_i = \eta_w \hat{p}_X(\mathbf{x}|\mathbf{w}_i, \sigma_i) \left(\frac{\mathbf{x}-\mathbf{w}_i}{\sigma_i^2} \right)$$

Width Update Rule

$$\Delta \sigma_i = \eta_w \hat{p}_X(\mathbf{x}|\mathbf{w}_i, \sigma_i) \cdot \frac{m}{\sigma_i} \left(\frac{\|\mathbf{x}-\mathbf{w}_i\|^2}{m\sigma_i^2} - 1 \right)$$

- $\hat{p}_X(\mathbf{x}|\mathbf{w}_i, \sigma_i)$ is the conditional posterior density of the i-th neuron
- Characterized by weight vector $\mathbf{w}_i$ and width σ_i
- η_w is the learning rate parameter

Winner-Take-All Approximation

Ideal Condition

Set conditional posterior density: $\hat{p}_X(\mathbf{x}|\mathbf{w}_j, \sigma_j) = \delta_{ji}$ for $j = 1, 2, \ldots, l$

where $\delta_{ji} = \begin{cases} 1 & \text{for } j = i \\ 0 & \text{for } j \neq i \end{cases}$

- When satisfied: neuron i is the **winning neuron**
- Conditional posterior density plays role of topological neighborhood function
- Setting $\hat{p}_X(\mathbf{x}|\mathbf{w}_j, \sigma_j) = h_{j,i(\mathbf{x})}$ connects to kernel SOM formulation

Key Connection

Update rules from KLD minimization have similar mathematical form to kernel SOM update rules!

Equivalence Statement

Fundamental Equivalence (Van Hulle, 2002b)

Minimization of the Kullback-Leibler divergence, assuming a Gaussian mixture model, is equivalent to maximization of the joint entropy defined in terms of incomplete gamma distribution kernels and an activity-based neighborhood function, which are at the core of kernel SOM.

Implications for Density Estimation

- Given data set $\{\mathbf{x}_k\}_{k=1}^{N}$
- Requirement: compute estimate of underlying distribution
- Distribution intrinsic to generation of the data
- Provides theoretical foundation for kernel SOM approach

Significance: Establishes theoretical connection between:

- Information-theoretic optimization (KLD minimization)
- Self-organizing neural networks (kernel SOM)

SOM Advantages and Applications

Key Advantages

- Topology preservation during dimensionality reduction
- Unsupervised learning capability
- Robust to noise and missing data
- Interpretable visual representations

Applications

- Data mining and exploratory analysis
- Pattern recognition and classification
- Vector quantization and data compression
- Visualization of high-dimensional data
- Modeling cortical organization

Summary

- SOMs provide powerful framework for unsupervised learning and visualization
- Two main approaches: elastic net visualization and contextual mapping
- Kernel SOMs address theoretical limitations through:
    - Principled kernel design using incomplete gamma distribution
    - Connection to information theory (KLD minimization)
    - Improved density estimation capabilities
- Wide range of applications from data compression to biological modeling
- Fundamental tool in modern machine learning and data analysis

Thank You!

Questions?