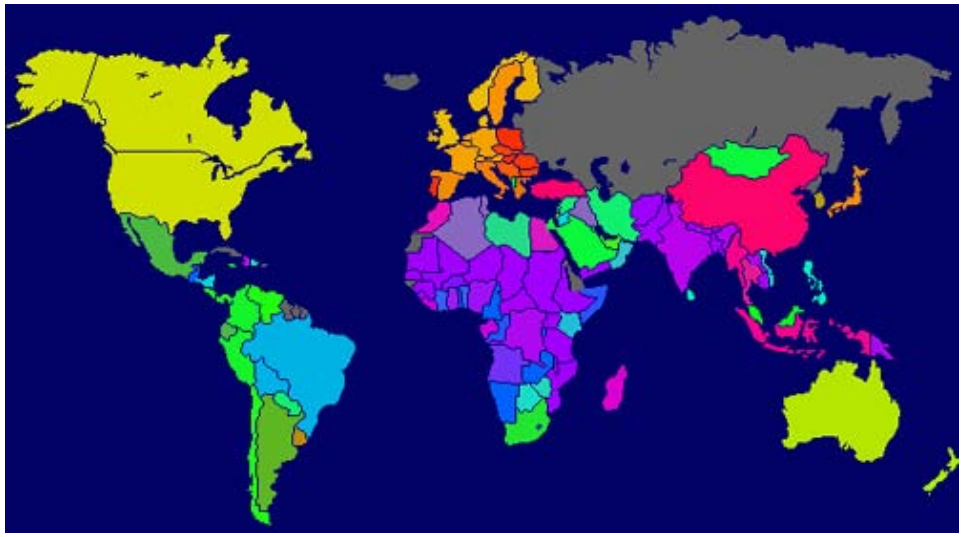
Unsupervised Learning: Self Organizing Maps

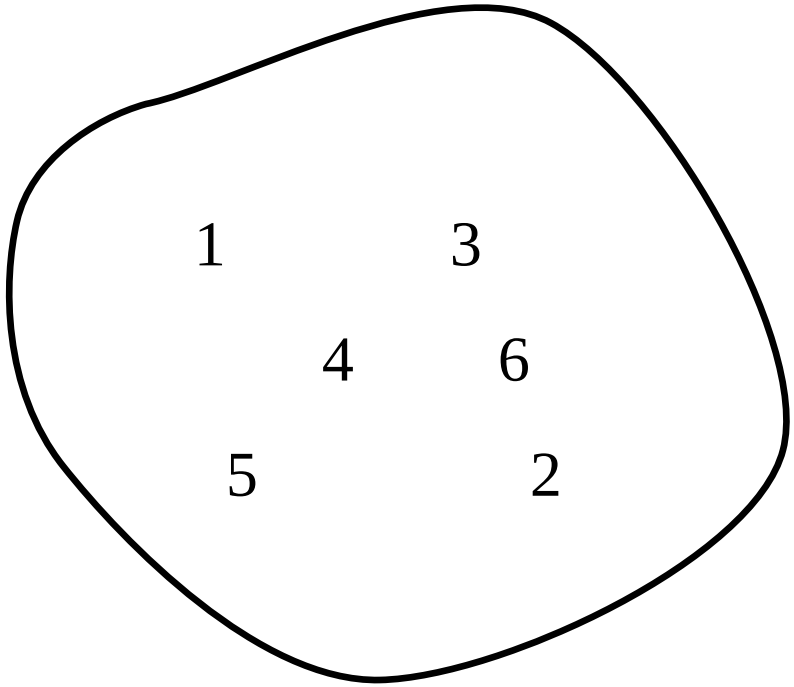


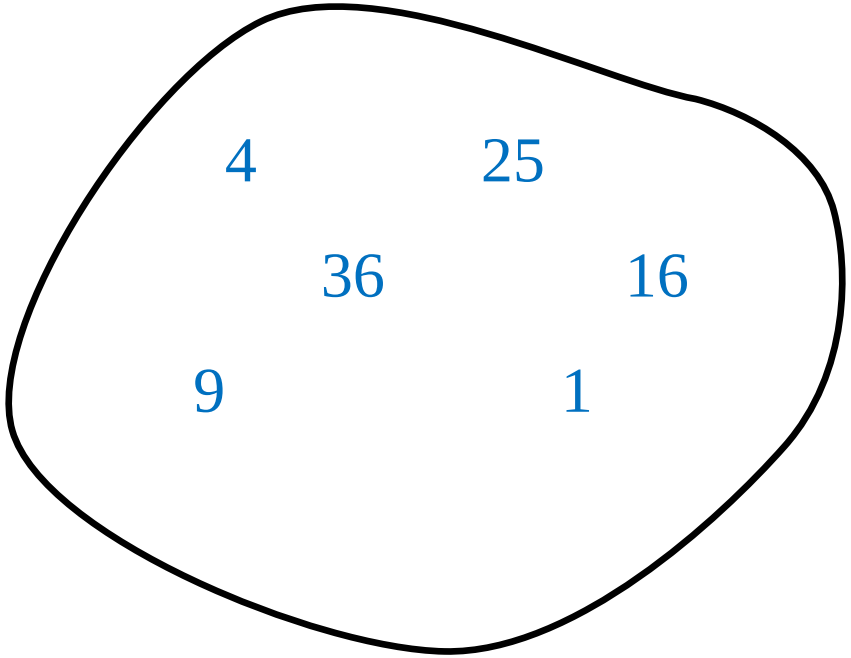
Jaeseung Jeong, Ph.D.

Department of Bio and Brain Engineering,

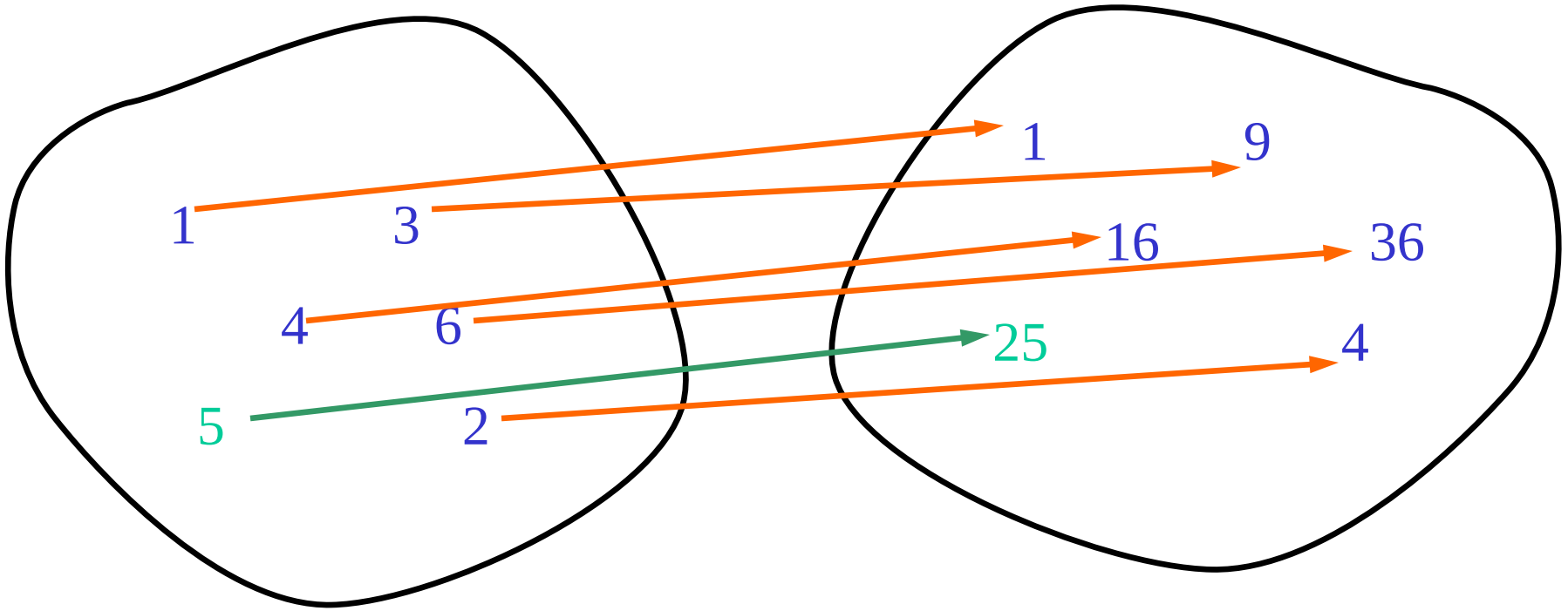
KAIST

Learning From Examples





Learning From Examples



Supervised Learning

- When ‘a set of targets of interest’ is provided by an external teacher, we say that the learning is **supervised.**

- The targets usually are in the form of an **input-output mapping** that the net should learn

Supervised Learning: Feed Forward Nets

- Feed Forward Nets learn under supervision
  - classification - all patterns in the training set are coupled with the “correct classification”
  - function approximation – the values to be learnt for the training points is known.

  - The Recurrent Networks also learn under supervision

Example: Hopfield Nets

- Associative Nets (Hopfield like) store predefined memories.
- During learning, the net goes over all patterns to be stored **(Hebb Rule)**:

$$W_{ij} = \frac{1}{N} \sum_{\mu} X_i^{\mu} X_j^{\mu}$$

Hopfield networks

- Hopfield Nets learn patterns whose organization is defined externally
- “Good” configurations of the system are the predefined memories

How do we learn in real life?

- Many times there is no “teacher” to tell us how to do things

How do we learn in real life?

- Many times there is no "teacher" to tell us how to do things
  - A baby that learns how to walk
  - Grouping of events (words of meaning) into a meaningful scene (making sense of the world)
  - Development of ocular dominance and orientation selectivity in our visual system

Self Organization

- **Network Organization** is fundamental to the brain
  - Functional structure: self-aggregation, clustering
  - Layered structure
  - Both ‘parallel processing and serial processing’ require organization of the brain

Self Organizing Networks

- Discover **significant patterns or features** in the input data
- Discovery is done **without a teacher**
- Synaptic weights are changed according to **‘local rules’**
- The changes affect a neuron’s immediate environment until **a final configuration** develops

Question

- How can a useful configuration develop from self organization?
- Can random activity produce coherent structure?

Answer: biologically

- There are self-organized structures in the brain
- Neuronal networks grow and evolve to be computationally efficient both *in vitro* and *in vivo*
- Random activation of the visual system can lead to layered and structured organization

Answer: Physically

- Local interactions can lead to global order
  - magnetic materials
  - Electric circuits
  - synchronization of coupled oscillators

Answer: Mathematically

- **A. Turing: Global order can arise from local interactions**
- Random local interactions between neighboring neurons can coalesce into states of global order, and lead to **coherent spatio- temporal behavior**

Answer: Mathematically

- Network organization takes place at 2 levels that interact with each other:
  - **Activity:** certain activity patterns are produced by a given network in response to input signals
  - **Connectivity:** synaptic weights are modified in response to neuronal signals in the activity patterns
- Self Organization is achieved if there is ***positive feedback*** between changes in synaptic weights and activity patterns

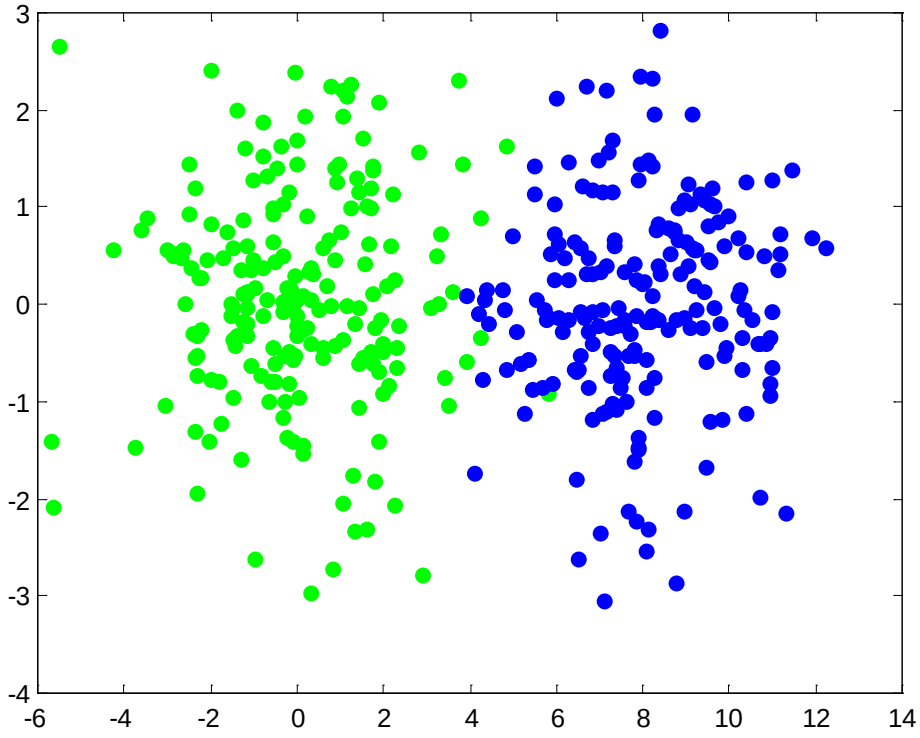
Principles of Self Organization

1. Modifications in synaptic weights tend to self amplify

2. Limitation of resources lead to competition among synapses

3. Modifications in synaptic weights tend to cooperate

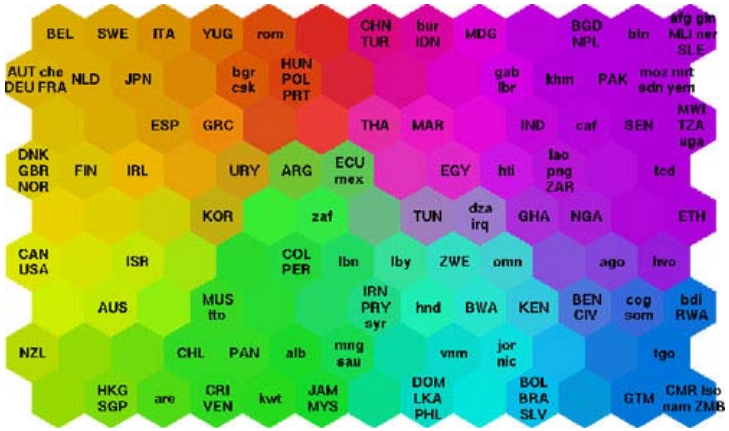
4. Order and structure in activation patterns represent **redundant information** that is transformed into knowledge by the network

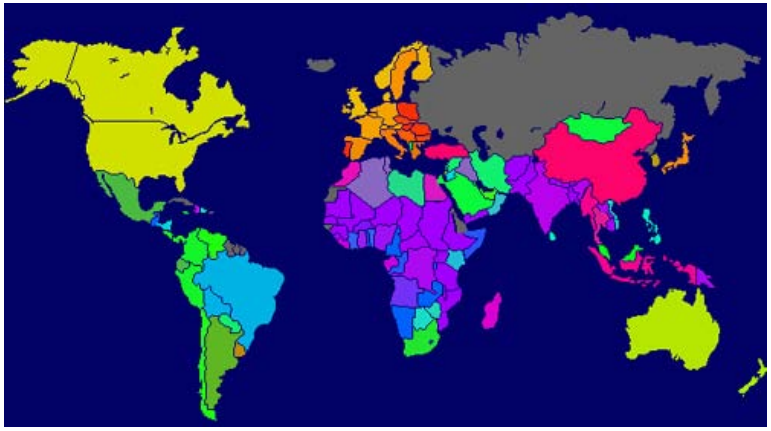

3
2
1
0
-1
-2
-3
-4
-6
-4
-2
0
2
4
6
8
10
12
14

Essential features of SOM

- A Self-Organizing Map (SOM) is a way to represent higher dimensional data in an usually 2-D or 3-D manner, such that similar data is grouped together.
- It runs unsupervised and performs the grouping on its own.
- Once the SOM converges, it can only classify new data. It is unlike traditional neural nets which are continuously learning and adapting.
- SOMs run in two phases:
  - Training phase: map is built, network organizes using a competitive process, it is trained using large numbers of inputs (or the same input vectors can be administered multiple times).
  - Mapping phase: new vectors are quickly given a location on the converged map, easily classifying or categorizing the new data.

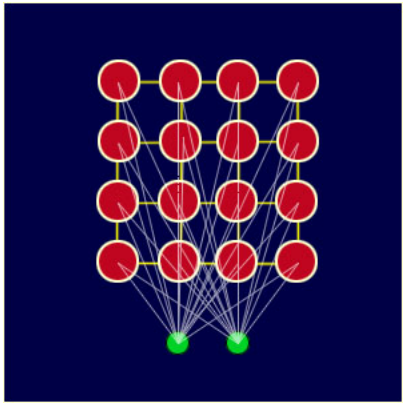
Uses

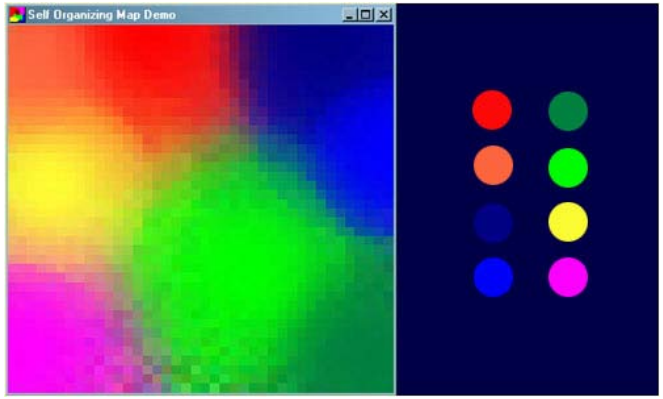


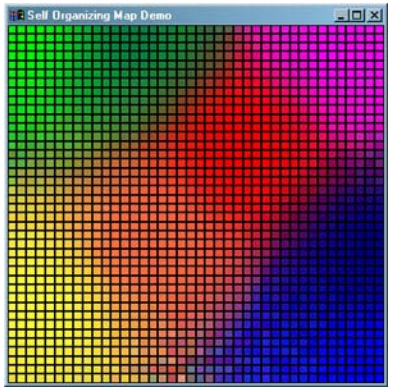


- Example: Data sets for poverty levels in different countries.
  - Data sets have many different statistics for each country.
  - SOM does not show poverty levels, rather it shows how similar the poverty sets for different countries are to each other. (Similar color = similar data sets).

SOM Structure







- Every node is connected to the input the same way, and no nodes are connected to each other.
- In a SOM that judges RGB color values and tries to group similar colors together, each square “pixel” in the display is a node in the SOM.
- Notice in the converged SOM above that dark blue is near light blue and dark green is near light green.

The Basic Process

1) Initialize each node's weights.
2) Choose a random vector from training data and present it to the SOM.
3) Every node is examined to find the Best Matching Unit (BMU).
4) The radius of the neighborhood around the BMU is calculated. The size of the neighborhood decreases with each iteration.
5) Each node in the BMU's neighborhood has its weights adjusted to become more like the BMU. Nodes closest to the BMU are altered more than the nodes furthest away in the neighborhood.
6) Repeat from step 2 for enough iterations for convergence.

Calculating the Best Matching Unit

- Calculating the BMU is done according to the Euclidean distance among the node's weights (W_1, W_2, … , W_n) and the input vector's values (V_1, V_2, … , V_n).
  - This gives a good measurement of how similar the two sets of data are to each other.

$$Dist = \sqrt{\sum_{i=0}^{i=n} (V_i - W_i)^2}$$

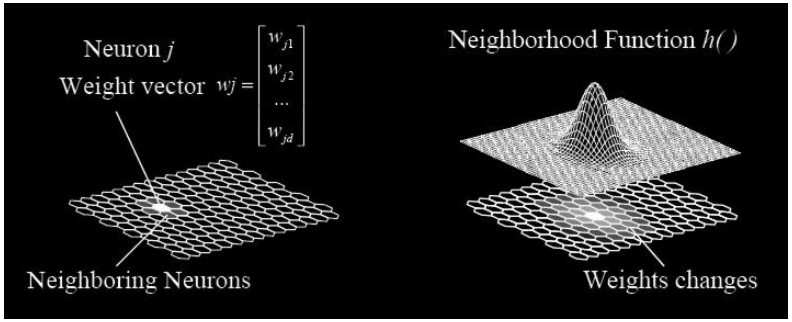
Determining the BMU Neighborhood

- Size of the neighborhood: We use an *exponential decay* function that shrinks on each iteration until eventually the neighborhood is just the BMU itself.

$$\sigma(t) = \sigma_0 \, exp\left(-\frac{t}{\lambda}\right)$$

- Effect of location within the neighborhood: The neighborhood is defined by a gaussian curve so that nodes that are closer are influenced more than farther nodes.

$$\Theta(t) = exp\left(-\frac{dist^2}{2\sigma^2(t)}\right)$$



Modifying Nodes' Weights

- The new weight for a node is the old weight, plus a fraction (L) of the difference between the old weight and the input vector… adjusted (theta) based on distance from the BMU.

$$W(t+1) = W(t) + \Theta(t) L(t) \left(V(t) - W(t)\right)$$

- The learning rate, L, is also an exponential *decay* function.
  - This ensures that the SOM will converge.

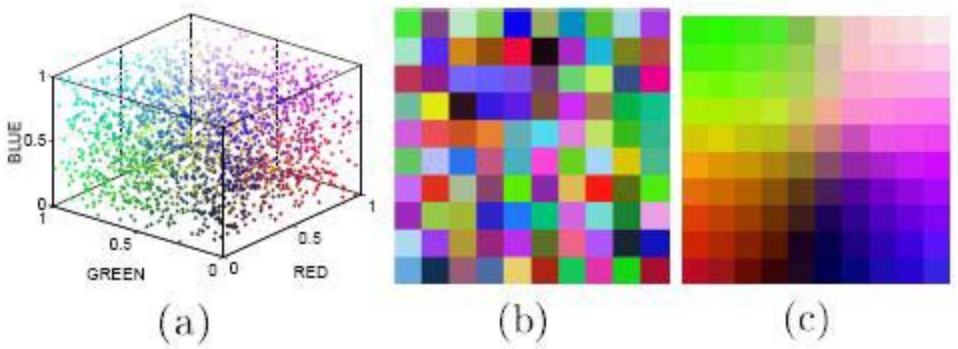
$$L(t) = L_0 \; exp\left(-\frac{t}{\lambda}\right)$$

- The lambda represents a time constant, and t is the time step

See It in Action

Example

- 2-D square grid of nodes.
- Inputs are colors.
- SOM converges so that similar colors are grouped together.
- Program with source code and pre-compiled Win32 binary: http://www.ai-junkie.com/files/SOMDemo.zip or mirror.



a) Input space
b) Initial weights
c) Final weights

Redundancy

- Unsupervised learning depends on redundancy in the data
- Learning is based on finding patterns and extracting features from the data

Types of Information

- **Familiarity** – the net learns how similar is a given new input to the typical (average) pattern it has seen before
- The net finds **Principal Components** in the data
- **Clustering –** the net finds the appropriate categories based on correlations in the data
- **Encoding –** the output represents the input, using a smaller amount of bits
- **Feature Mapping –** the net forms a topographic map of the input

Possible Applications

- Familiarity and PCA can be used to analyze unknown data
- PCA is used for dimension reduction
- Encoding is used for vector quantization
- Clustering is applied on any types of data
- Feature mapping is important for dimension reduction and for functionality (as in the brain)

Simple Models

- Network has inputs and outputs
- There is no feedback from the environment ➔ no supervision
- The network updates the weights following some learning rule, and finds patterns, features or categories within the inputs presented to the network

Unsupervised Hebbian Learning

- One linear unit: $V = \sum_j W_j X_j$

  Hebbian Learning

$$\Delta W_j = \eta V X_j$$

Problems:

- In general W is not bounded
- Assuming it is bounded, we can show that it is not stable.

Oja's Rule

- The learning rule is Hebbian like:

$$\Delta W_j = \eta V (X_j - V W_j)$$

The change in weight depends on the product of the neuron's output and input, with a term that makes the weights decrease

Alternatively, we could have normalized the weight vector after each update, keeping its norm to be one.

Oja's Rule, Cntd

- Such a net converges into a weight vector that:
  - Has norm = 1
  - Lies in the direction of the maximal eigenvector of C (correlation matrix of the data)
  - Maximizes the (average) value of ^{2}

Oja's Rule, Cntd

- This means that the weight vector points at the first **principal component** of the data
- The network learns a feature of the data without any prior knowledge
- This is called **feature extraction**

Visual Model

Linsker (1986) proposed a model of self organization in the visual system, based on unsupervised Hebbian learning

- Input is random dots (does not need to be structured)
- Layers as in the visual cortex, with FF connections only (no lateral connections)
- Each neuron receives inputs from a well defined area in the previous layer (“receptive fields”)
- The network developed **center surround** cells in the 2nd layer of the model and **orientation selective** cells in a higher layer
- A self organized structure evolved from (local) hebbian updates