Neural Networks: Self-Organizing Networks

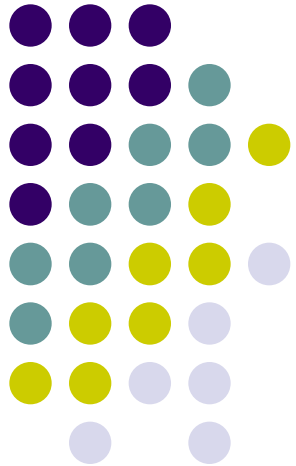
Prof. dr. sc. Sven Lončarić

Doc. dr. sc. Tomislav Petković

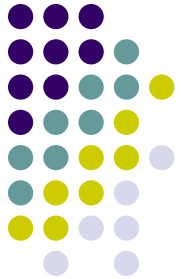
University of Zagreb

Faculty of Electrical Engineering and Computing

http://www.fer.unizg.hr/en/course/neunet_a

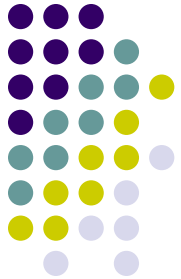


Introduction



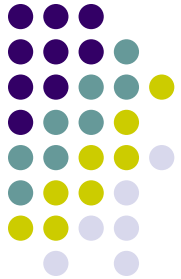
- Self-organizing neural networks learn without supervision
- In unsupervised learning
  - there is no external teacher
  - there are no correct answers, i.e. there are no labels for the data nor is there a direct reward for positive behavior
- The network must discover underlying data structure such as patterns, correlations and categories by itself
- Unsupervised learning is only possible if learning data exhibits redundancy
  - Without redundancy the input data resembles random noise making learning impossible

Introduction



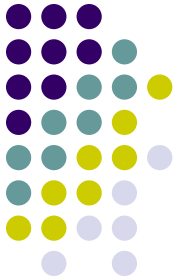
- The type of underlying structure the self-organizing network may infer in the input data depends on the network architecture
- There are may different architectures of self-organizing networks which are used for different purposes
- In general, the network must learn to represent the input and must produce some output based on the learned representation
  - What are the outputs? What do they represent?

What do outputs represent?



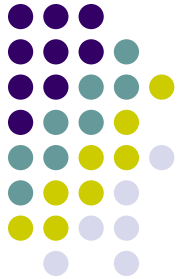
1. Similarity: There is a single output showing a similarity measure between the input sample and all previous (averaged) samples.
2. Principal Component Analysis (PCA): Extension of similarity measure to multiple vectors, i.e. network outputs similarities to the eigenvectors of the data correlation matrix.
3. Clustering: The output is a binary (indicator) vector where only one bit is active, i.e. the network discovers the inherent groupings in the data.

What do outputs represent?



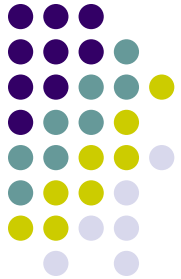
4. Prototyping: Similar to clustering except the output is a representative sample of a cluster instead of an indicator function (e.g. associative memory).
5. Coding: The output is a coded version of the input using less bits while keeping most of information content intact (e.g. vector quantization, data compression).
6. Feature Mapping: The output neurons are arranged on a regular grid (e.g. a rectangular lattice) and only one may be active at any single moment.

Lecture Overview



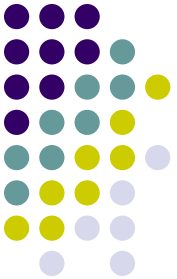
- Unsupervised Hebbian learning
  - A single self-organizing neuron
  - Oja's learning rule
  - Principal component analysis (PCA)
  - Single layer self-organizing feedforward network
  - Sanger's learning rule (GHA)
- Competitive learning without supervision
- Kohonen's self-organizing map

A Self-Organizing Linear Neuron



- Let us consider a simple case of a single linear neuron
- Let ξ be a N-dimensional random vector whose components are ξ_i, $i = 1,\ldots, N$
- A single self-organizing linear neuron learns by observing a large collection of input samples ξ
- After observing a sufficient number of samples the neuron should output a similarity measure v indicating how well a particular input ξ matches the distribution of inputs

A Self-Organizing Linear Neuron



- The output v of a single linear neuron is

$$v = \sum_{j=1}^{N} w_j \xi_j = \mathbf{w}^T \boldsymbol{\xi} = \boldsymbol{\xi}^T \mathbf{w}$$

where $\mathbf{w}$ is the vector of input weights

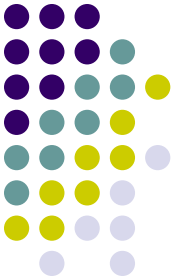
- The output (or activation potential) v should have higher values for more probable inputs $\boldsymbol{\xi}$
- To achieve this we may use Hebbian learning

$$\Delta w_i = w_i[n+1] - w_i[n] = \eta v \xi_i = \eta \underbrace{\left(\mathbf{w}^T[n]\,\boldsymbol{\xi}\right)}_{\text{a number}} \xi_i$$

which in a matrix form becomes

$$\Delta \mathbf{w} = \mathbf{w}[n+1] - \mathbf{w}[n] = \eta \underbrace{\boldsymbol{\xi}\boldsymbol{\xi}^T}_{\text{a } N \times N \text{ matrix}} \mathbf{w}[n]$$

Training with the Hebbian Rule



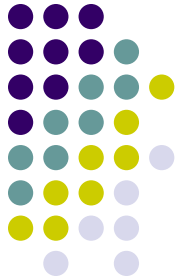
- The weights are the updated using the following recurrence equation

$$\mathbf{w}[n+1] = \mathbf{w}[n] + \eta\, \xi\xi^T\, \mathbf{w}[n] = \left(\mathbf{I} + \eta\, \xi\xi^T\right) \mathbf{w}[n]$$

- This equation describes an unstable dynamical system
- The basic Hebbian scheme leads to an unrealisitc growth of the weights
  - A normalization or saturation may be used to limit the growth of weights

$$w_i[n+1] = \frac{w_i[n] + \eta(\mathbf{w}^T\xi)\xi_i}{\left(\sum_{i=1}^{N}\left|w_i[n] + \eta(\mathbf{w}^T\xi)\xi_i\right|^p\right)^{1/p}}$$

Oja's Learning Rule



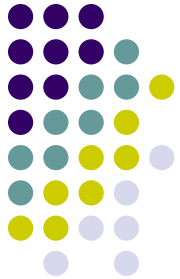
- Proposed by Erkki Oja in 1982
- Provides a simple recurrent update equation which (under certain constraints) has stable fixed points on the unit circle
- Oja's learning rule for unsupervised learning is

$$\Delta w_i = w_i[n+1] - w_i[n] = \eta v \left(\xi_i - \underbrace{v w_i[n]}_{\text{forgetting term}}\right)$$

or expressed in a matrix form

$$\Delta \mathbf{w}_i = \mathbf{w}_i[n+1] - \mathbf{w}_i[n] = \eta \left(\boldsymbol{\xi}\boldsymbol{\xi}^T - \mathbf{w}^T[n]\, \boldsymbol{\xi}\boldsymbol{\xi}^T\, \mathbf{w}[n]\right) \mathbf{w}[n]$$

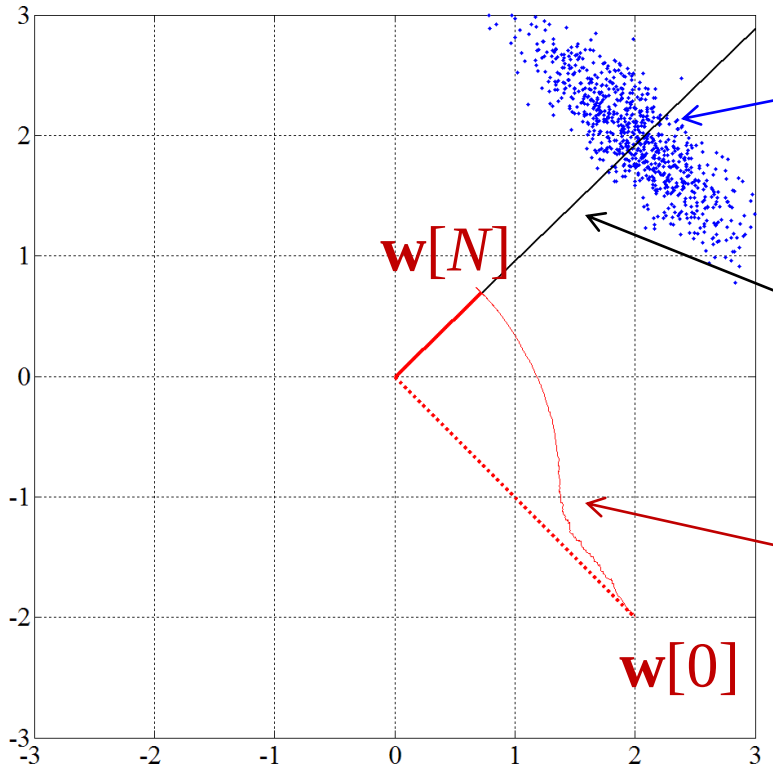
Properties of Oja's Rule



- Under certain limitations on the learning rate η the Oja's rule produces weights $\mathbf{w}[n]$ with the following asymptotic ($n \to \infty$) behavior:

1. $\left|\mathbf{w}[n]\right| \to 1$
2. The direction of the synaptic vector $\mathbf{w}[n]$ tends to the direction of the *dominant eigenvector* of the input correlation matrix $\mathbf{C} = \mathbb{E}[\boldsymbol{\xi}\boldsymbol{\xi}^T]$.
3. $\mathbf{w}[n]$ is such that activation potential $v = \mathbf{w}^T[n]\,\boldsymbol{\xi}$ has the largest quadratic mean, i.e. it tends to the direction which maximizes $\mathbb{E}[v^2]$.

Example of Oja's Rule for non-centered data



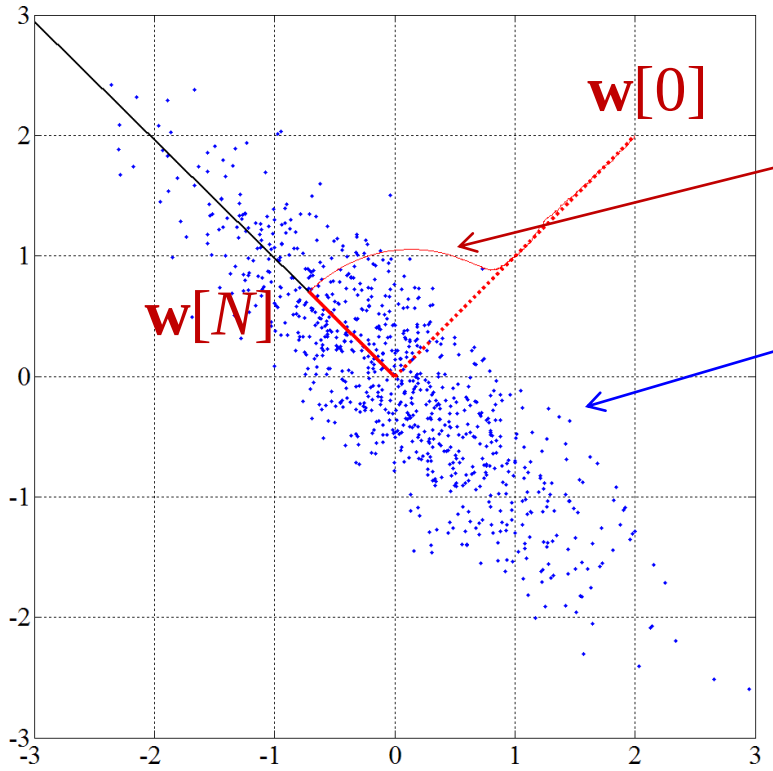
Normally distributed two-dimensional samples

The weights $\mathbf{w}[N] = (3.100\text{E+}50,\ 2.9856\text{E+}50)$ for the Hebbian learning are *unconstrained*, however the direction is the same

Trajectory of weights during unsupervised learning using $\eta = 0.001$

- A total of $N = 15000$ samples ξ are drawn from a multivariate Gaussian distribution with $\sigma = 0.45$, $\rho = -0.85$, and mean at $(2, 2)$
- The starting weight is $\mathbf{w}[0] = (-2, -2)$
- The ending weight is $\mathbf{w}[N] = (0.7204, 0.6937)$

Example of Oja's Rule for centered data

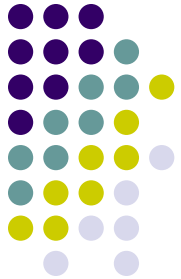


Trajectory of weights during unsupervised learning using $\eta = 0.001$

Normally distributed and *centered* two-dimensional samples.

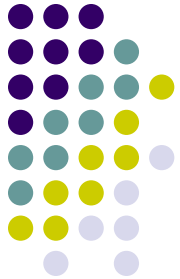
- A total of $N = 15000$ samples ξ are drawn from a multivariate Gaussian distribution with $\sigma = 0.85$, $\rho = -0.85$, and mean at $(0, 0)$
- The starting weight is $\mathbf{w}[0] = (2, 2)$
- The ending weight is $\mathbf{w}[N] = (-0.7140, 0.7002)$

Principal Component Analysis



- Proposed by Karl Pearson 1901
- Formalized (and named) by Harold Hotelling in 1933
- It is a well known method in statistical analysis
  - It converts a set of observations into a set of values of linearly uncorrelated variables
- Has several other names
  - *Eigenvalue decomposition* of $\mathbf{X}^T\mathbf{X}$ in linear algebra (where $\mathbf{X}$ is a data matrix containing one sample $\mathbf{x} = \boldsymbol{\xi}^T$ per *row*)
  - *Karhunen-Loève transform* (KLT) in signal processing

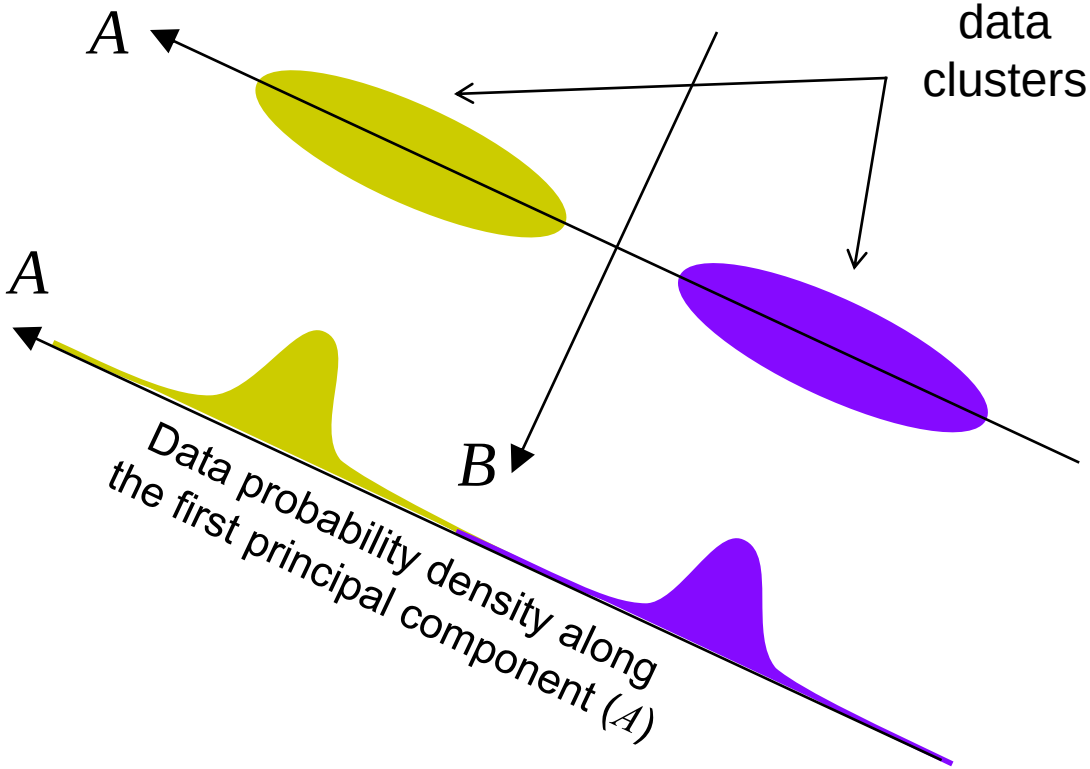
Principal Component Analysis

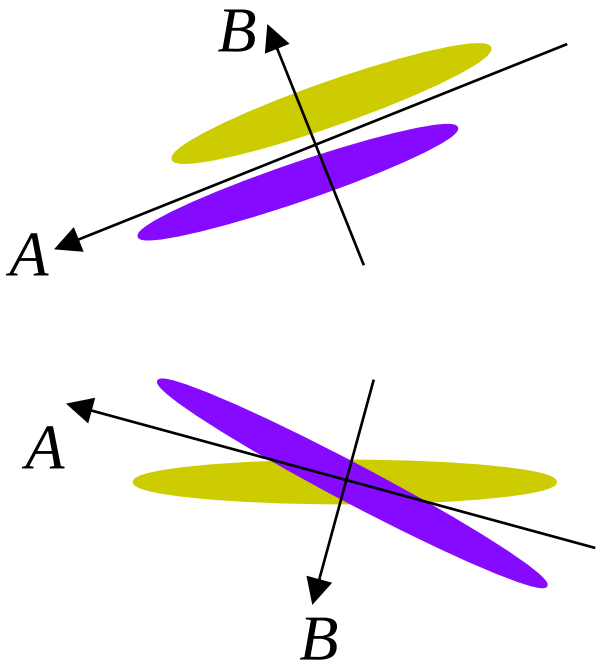


- The basic idea of PCA is to use an orthogonal transform which converts a set N-dimensional observations into a set of M values which are linearly uncorrelated
  - New M values are called principal components
  - Transformation is defined so the first principal component has the largest possible variance
- The number of principal components M is less than or equal to N, i.e. $M \leq N$
  - Usually $M << N$ hence the transform performs data reduction which may make certian data analysis tasks simpler

Principal Component Analysis

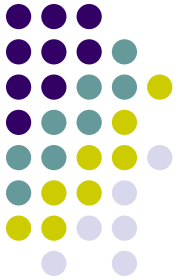
- For some datasets it is easier to recognize and separate clusters when the data is projected on the axis of the largest variance A than when projected on the orthogonal axis B





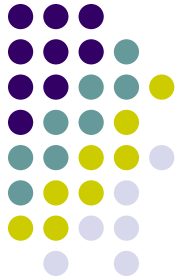
- For other datasets such property may not hold

Input Data Correlation Matrix



- Let $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_N]^T$ be a N-dimensional random vector
  - Its components ξ_i are random variables
- Then the correlation matrix of $\boldsymbol{\xi}$ is $\mathbf{C} = \mathbb{E}[\boldsymbol{\xi}\boldsymbol{\xi}^T]$
  - Each element c_{ij} of $\mathbf{C}$ is a correlation between a pair of random variables ξ_i and ξ_j, i.e. $c_{ij} = \mathbb{E}[\xi_i \xi_j]$
  - The diagonal elements of $\mathbf{C}$ are variances of individual components of $\boldsymbol{\xi}$, i.e. $c_{ii} = \mathbb{E}[\xi_i^2]$
  - $\mathbf{C}$ is a *symmetrical* matrix, i.e. $c_{ij} = c_{ji}$
  - $\mathbf{C}$ is a *positive-semidefinite* or *Grammian* matrix, i.e. $\mathbf{x}^T\mathbf{C}\mathbf{x} \geq 0$ for all $\mathbf{x}$
  - Do not confuse $\mathbf{C}$ with a *Pearson correlation matrix* from statistics which is a matrix containing Pearson product-moment correlation coefficients

Covariance Matrix

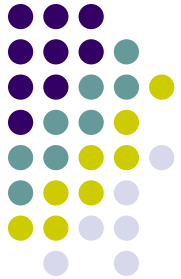


- The covariance matrix of a random vector $\boldsymbol{\xi}$ is

$$\boldsymbol{\Sigma} = \mathbb{E}\left[(\boldsymbol{\xi} - \mathbb{E}[\boldsymbol{\xi}])(\boldsymbol{\xi} - \mathbb{E}[\boldsymbol{\xi}])^T\right]$$

- Each element of $\boldsymbol{\Sigma}$ is a covariance between a pair of random variables ξ_i and ξ_j
- The covariance matrix $\boldsymbol{\Sigma}$ is *symmetrical* and *positive-semidefinite*
- If $\boldsymbol{\xi}$ is centered, i.e. $\mathbb{E}[\boldsymbol{\xi}] = 0$, then correlation matrix and covariance matrix are equal, $\mathbf{C} = \boldsymbol{\Sigma}$
  - This condition is sometimes/often implicitly assumed but not explicitly stated

Principal Component Analysis



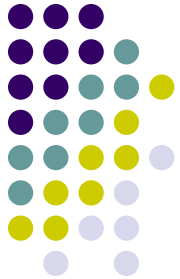
- Claim:

  The direction of the maximal variance of data is the direction of the first main component, i.e. of the eignevector corresponding to the maximal absolute eigenvalue of the covariance matrix Σ

- Sketch of the proof:

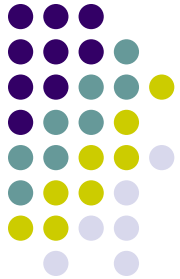
  The claim follows from the fact that Σ is positive-semidefinite by analysing the associated quadratic form $\mathbf{x}^T\Sigma\mathbf{x}$ for all real vectors $\mathbf{x}$. First we show the quadractic form is a strictly convex function. Then we show the quadratic form $\mathbf{x}^T\Sigma\mathbf{x}$ obtains the largest minimum when $\mathbf{x} = k\mathbf{v}_{\text{dom}}$, $k \in \mathbb{R}$, i.e. $\mathbf{x}$ is in the direction of the dominant eigenvector $\mathbf{v}_{\text{dom}}$.

Principal Component Analysis



- It may be easily demonstrated that the second principal component must be orthogonal to the frist one and must again represent the first principal component in a $(N-1)$-dimensional subspace where the first principal component was removed
- The second principal comopnent is then equal to the second dominant eigenvector of the covariance matrix
- Repeating this argument yields complete singal representation in the form of its principal components

Principal Component Analysis and Oja's Learning Rule

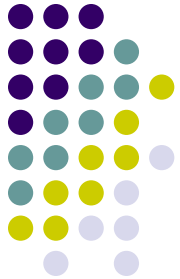


- What about a self-organizing single linear neuron trained using Oja's rule?
- We have

$$\mathbb{E}[v^2] = \mathbb{E}\left[(\mathbf{w}^T\boldsymbol{\xi})^2\right] = \mathbb{E}\left[\mathbf{w}^T\boldsymbol{\xi}\boldsymbol{\xi}^T\mathbf{w}\right] = \mathbf{w}^T\,\mathbf{C}\,\mathbf{w}$$

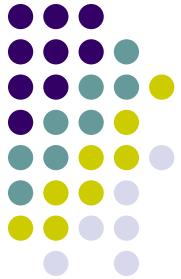
- Hence a single neuron tranied using Oja's rule outputs the value of the first principal component of the data if the data is centered (so $\mathbf{C}$ equals $\boldsymbol{\Sigma}$)
  - If the input data is not centered then the output of a neuron trained using Oja's rule is not the first principal component

Principal Component Analysis



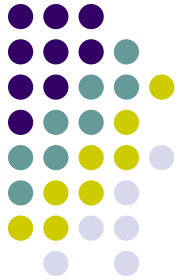
- Let $\mathbf{A}$ be an orthogonal matrix, i.e. $\mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T = \mathbf{I}$, whose rows are orthonormal eigenvectors of the correlation matrix $\mathbf{C} = \mathbb{E}[\boldsymbol{\xi}\boldsymbol{\xi}^T]$
- A linear transformation of ξ into a new vector $\psi = \mathbf{A}\xi$ projects the data from the original space into a new space of principal components
- Orthonormal eigenvectors show the directions of the principal components
  - If eigenvectors are ordered in the descending absolute values of corresponding eigenvalues then data variation along each successive eigenvector decreases

Principal Component Analysis: Dimensionality Reduction



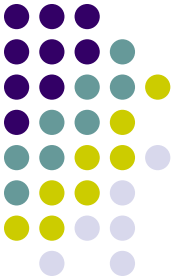
- If we keep only the first $M < N$ components of the vector ψ then such reduced vector ψ_M is an N-dimensional representation of ξ which minimizes the mean squared error (MSE), i.e. $\mathbb{E}[\ \|\boldsymbol{\xi} - \mathbf{A}^T\psi_M\|_2^2\ ]$ is minimal
- Every other linear transformation which reduces the number of components to M will have a larger MSE
- The reduced vector ψ_M may significantly simplify some common data analysis tasks such as classification and/or clustering of original samples ξ

Single Layer Feedforward Network for Optimal Signal Decomposition



- Oja'a rule is applicable to a single neuron only
  - Can it be extended to a single layer feedforward network comprised of M linear neurons?
- Sanger proposed in 1989 to combine Oja's rule and Gram-Schmidt orthogonalization process
  - He also gives a formal proof of convergence
- Sanger's network decomposes the input using *eigenvector decomposition*, i.e. it produces a Karhunen-Loève transform (KLT) of the input

Sanger's Rule
Generalized Hebbian Algorithm - GHA

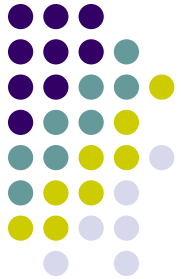


- Consider a single-layer network comprised of M linear neurons admitting an N dimensional input
  - $N >> M$ usually holds, i.e. we want the network to preform dimensionality reduction
- The output v_i of ith neuron is

$$v_i = \sum_{j=1}^{N} w_{ij}\xi_j = \mathbf{w}_i^T \boldsymbol{\xi} = \boldsymbol{\xi}^T \mathbf{w}_i$$

  where $\mathbf{w}_i$ is N-dimensional weight vector and where $\boldsymbol{\xi}$ is N-dimensional input vector
- How do we learn the weights to get PCA or KLT?

Sanger's Rule
Generalized Hebbian Algorithm - GHA



- Oja's rule for updating jth weight of one neuron is:

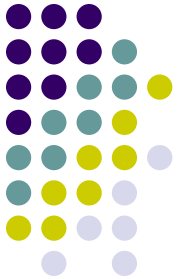
$$\Delta w_j = w_j[n+1] - w_j[n] = \eta v\left(\xi_j - v w_j[n]\right)$$

- Sanger's rule for updating jth weight of ith neuron in a single-layer network is:

$$\Delta w_{ij} = w_{ij}[n+1] - w_{ij}[n] = \eta v_i\left(\xi_j - \sum_{k=1}^{i} v_k w_{kj}[n]\right)$$

- For the first neuron ($i = 1$) Sanger's rule is the same as Oja's rule, hence weights $w_{1j}[n]$ converge to the first eigenvector of the input correlation matrix
- Sanger's rule is also known as generalized Hebbian algorithm or GHA

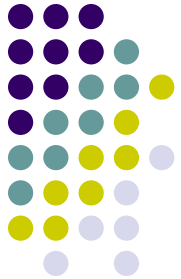
Sanger's Rule
Generalized Hebbian Algorithm - GHA



- Each of M linear neurons produces the output v_i which is the decomposition of the input sample ξ using the basis vectors $\mathbf{w}_i[n]$
- If input samples ξ are centered, i.e. $\mathbb{E}[\xi] = 0$, then:
  1. The outputs v_i , $i = 1,\ldots, M$, are the first M principal components of ξ
  2. The weights $\mathbf{w}_i[n]$ define the loading vectors
  3. The input sample ξ may be aproximated using only M components (with the minimal total squared error) as

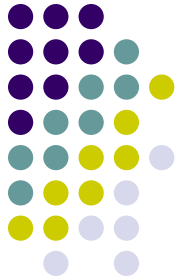
$$\xi \approx \sum_{i=1}^{M} v_i \mathbf{w}_i[n]$$

Competitive Learning



- In Hebbian learning multiple neurons may be simultaneously active
- In competitive learning only a single neuron may be active
  - In larger networks we may relax this and require that a single neuron in a group is active
- Networks having such properties are called winner-take-all or WTA networks
- The purpose of such networks is grouping or categorization of data, i.e. similar inputs are classified into the same group
  - They are often used for pattern classification in computer vision or for vector quantization

Winner-Take-All Neural Networks



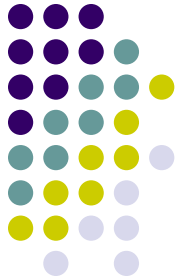
- Consider a simple WTA neural network comprised of a single fully-connected feedforward layer, i.e. all inputs are fed to every output neuron
- Only one neuron may be activated (the winner)
  - The winning neuron is also called best matching unit or BMU
- The winner is a neuron with the highest activation potential:

$$k = \arg\max_i \sum_j w_{ij}\xi_j = \arg\max_i \mathbf{w}_i^T \boldsymbol{\xi}$$

- Hence the activation potential v_k of the winning neuron satisfies the inequality:

$$v_k = \mathbf{w}_k^T \boldsymbol{\xi} \geq \mathbf{w}_i^T \boldsymbol{\xi} = v_i \quad \forall i$$

Winner-Take-All Neural Networks

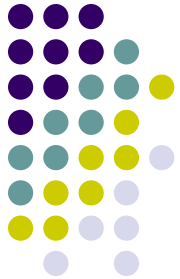


- If the weights are normalized to unity so $\|\mathbf{w}_i\| = 1$ then the previous inequality may be transformed to:

$$\left\|\mathbf{w}_k - \boldsymbol{\xi}\right\| \leq \left\|\mathbf{w}_i - \boldsymbol{\xi}\right\| \quad \forall i$$

- In other words the winner is a neuron whose input weights $\mathbf{w}_i$ are closest to the input sample $\boldsymbol{\xi}$
- How do we learn the weights $\mathbf{w}_i$ for a WTA network?
- One possible learning strategy is to use a standard competitive learning rule

Competitive Learning



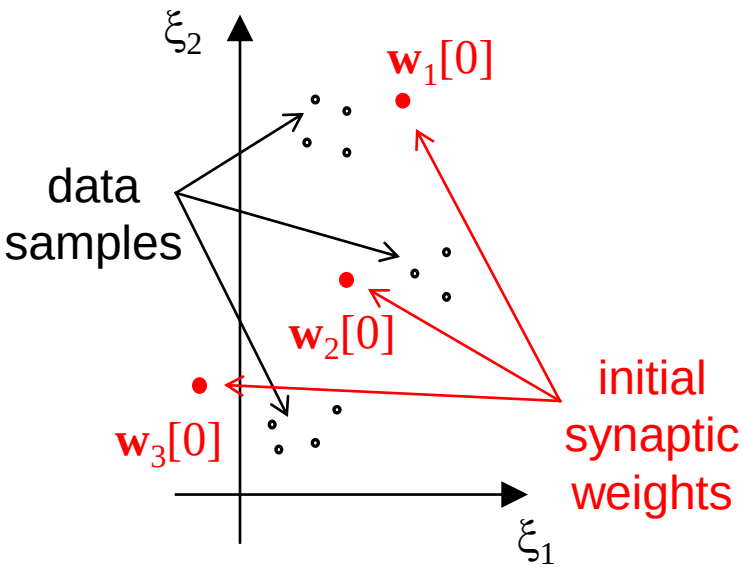
- The vectors of synaptic weights are randomly initialized
- We sequentially feed the network with data samples in randomized order
- For each data sample $\boldsymbol{\xi}$ we determine which neuron is the winner and we modify only the weights of that kth neuron using the following rule:

$$\Delta w_{kj} = w_{kj}[n+1] - w_{kj}[n] = \eta(\xi_j - w_{kj}[n])$$

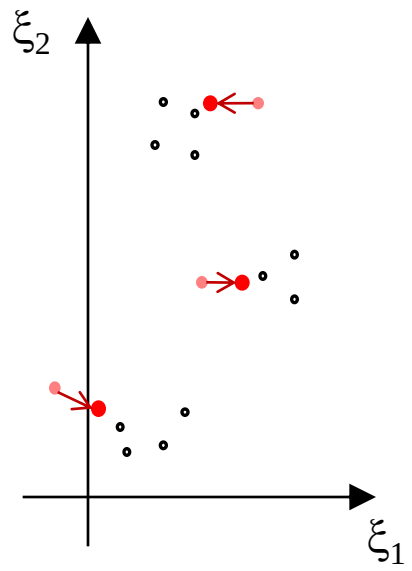
- This standard competitive learning rule shifts the weight vector $\mathbf{w}_k$ of the winning neuron in the direction of the input vector $\boldsymbol{\xi}$

Competitive Learning: Example

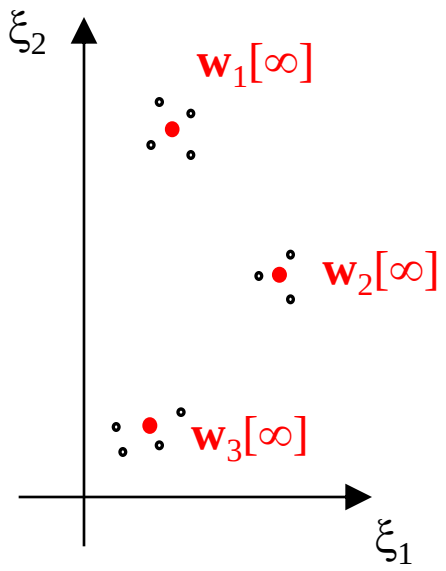
- Consider a network of three neurons arranged in a single layer which admits two-dimensional input $\boldsymbol{\xi} = [\xi_1\ \xi_2]^T$



initial synaptic weights $\mathbf{w}_i[0]$ are randomly initialized

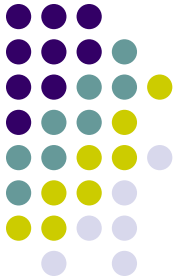


intermediate state during training



the final state after convergence

Assesing the Quality of Grouping



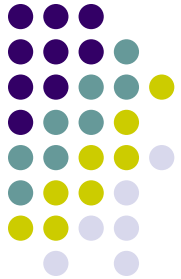
- It is often necessary to asses the quality of grouping when using unsupervised learning
- The standard competitive learning rule has an associated cost function:

$$E = \frac{1}{2}\sum_{l=1}^{L}\sum_{i=1}^{C}\sum_{j=1}^{N} m_{il}(\xi_j^l - w_{ij})^2$$

where L is the number of vectors, C is the number of groups, N is the number of dimensions, and $\mathbf{M} = (m_{il})$ is an indicator matrix showing to which group G_i the sample ξ^l belongs:

$$m_{il} = \begin{cases} 1, & \xi^l \in G_i \\ 0, & \xi^l \notin G_i \end{cases}$$

Assesing the Quality of Grouping



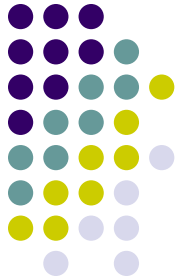
- An equivalent form of the cost function which shows the quality of grouping is:

$$E = \frac{1}{2}\sum_{l=1}^{L}\left\|\boldsymbol{\xi}^{l} - \mathbf{w}_{k(l)}\right\|^{2}$$

  where $k(l)$ is the index of the winning neuron for the l-th input sample $\boldsymbol{\xi}^{l}$

- The cost is a sum of squared distances between the input samples and the centers of groups to which they belong
- It is a measure which is lower the closer the input vectors $\boldsymbol{\xi}$ are to the group center vectors $\mathbf{w}_k$

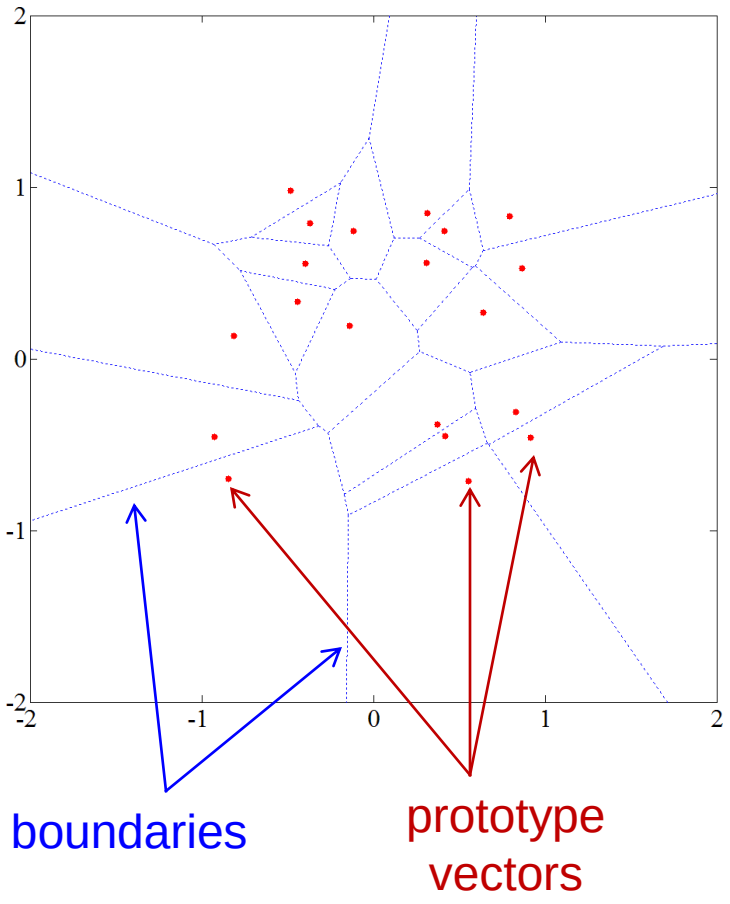
Example: Vector Quantization



- The problem is to partition the input set of vectors into M partitions and then to represent each of M partitions using one (optimal) prototype vector
  - A set of M representative vectors is called a code book
- If M is small we achieve data compression
  - We store and transmit only indices of prototype vectors
  - We also need to transmit the code book, but only once
- Optimal representation may be achieved by requiring e.g. that representative vector is closest to all samples of its partition
  - Such choice produces a Voronoi tessellation of input space

Voronoi Tessellation

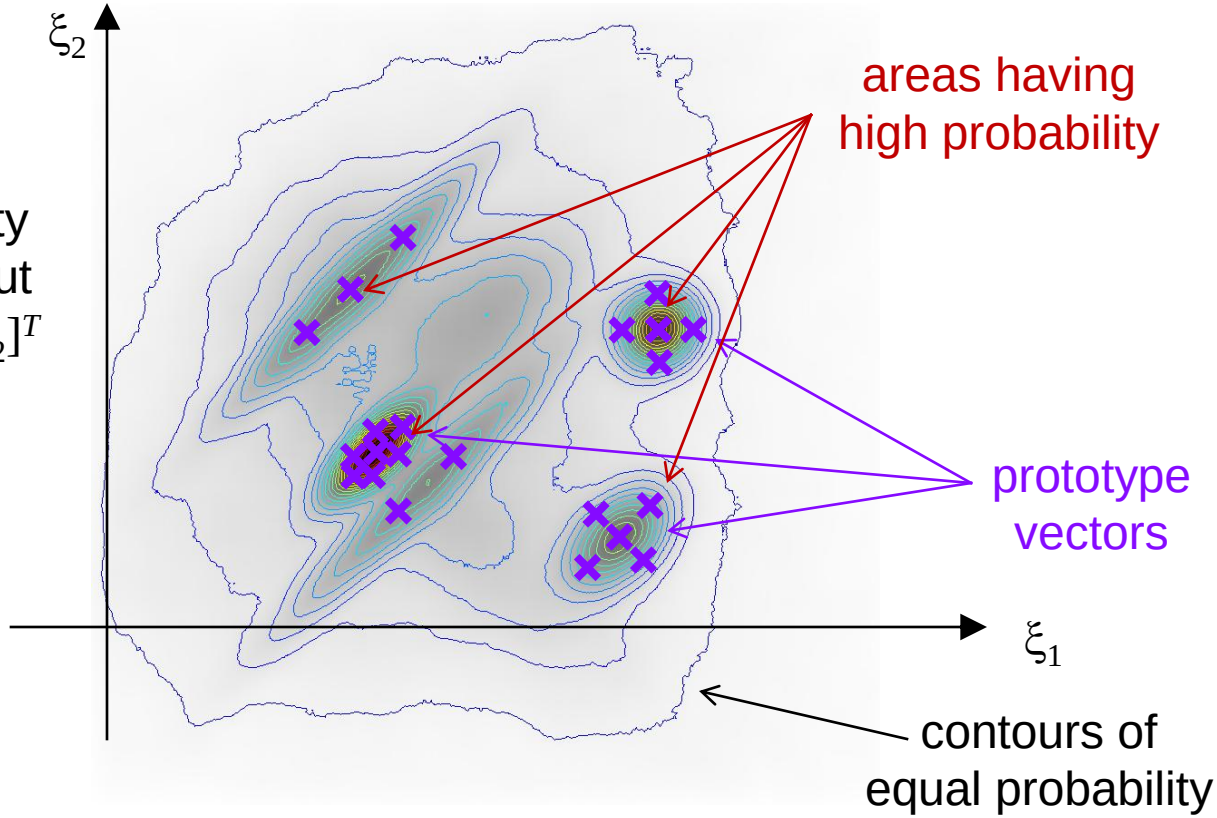
- In Voronoi tesselation the space is partitioned into disjoint polygonal regions which are defined by associated prototype vectors
- Each polygonal region is a set of points which are closest (in the sense of Euclidean distance) to the associated prototype vector
- Boundaries between polygonal regions are orthogonal to the lines connecting prototype vectors



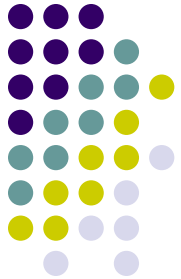
Example: Vector Quantization

- An example of using competitive learning to desing a code book of 22 prototype vectors for two dimensional data

probability density function over input samples $\boldsymbol{\xi} = [\xi_1\ \xi_2]^T$

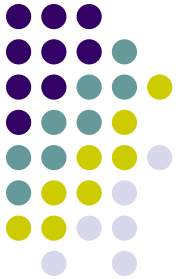


Feature Mapping



- Previous examples do not take into consideration the spatial arrangement of output layer neurons
- It is often useful to arrange the output layer neurons into a regular lattice
- We can then define neighborhoods on such regular lattices and modify the competitive learning rule to require that neurons which are close in the output layer respond to samples which are close in the input space
- This defines a mapping which preserves topological properties of the input space
- It is especially useful in visualizing high-dimensional data in one or two dimensions

Kohonen Map or SOM



- Proposed by Teuvo Kohonen in 1982
- Kohonen's learning rule is modified standard competitive learning rule

$$\Delta w_{ij} = \eta \Lambda(i,k)(\xi_j - w_{ij})$$

where $\Lambda(i,k)$ is a function which decreases with the distance from the winning neuron k and which determines the neighborhood in which the weight update is performed

Example of 2D Kohonen Map

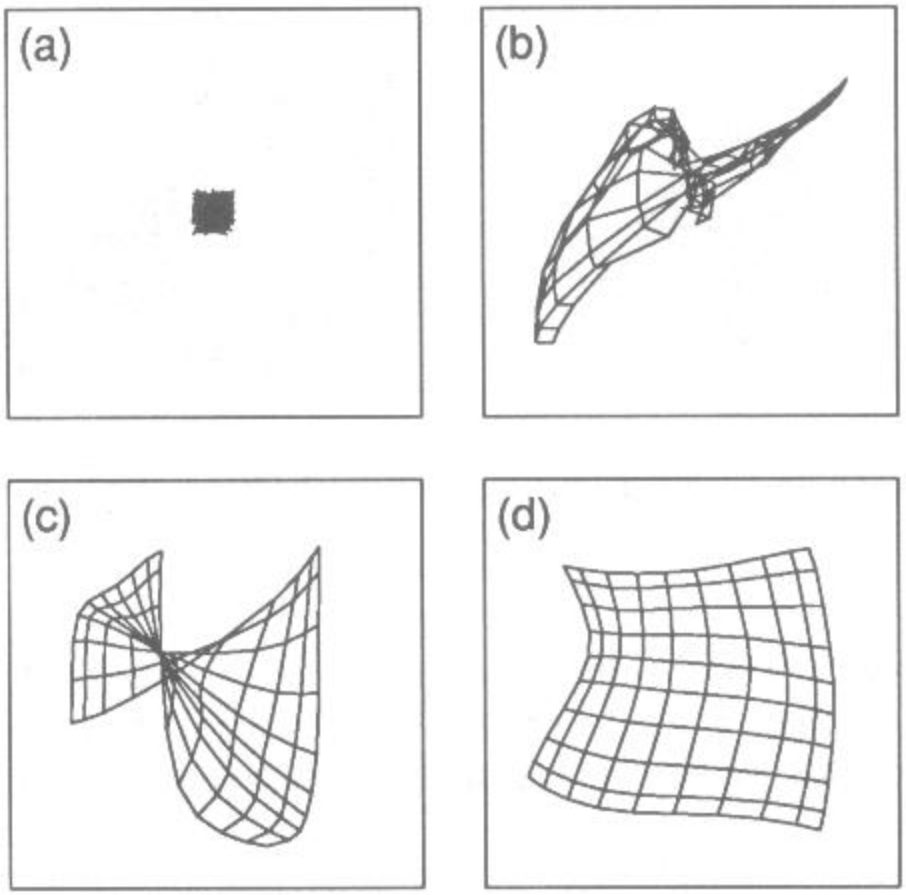
- We train a Kohonen map of 100 neurons arranged in a regular rectangular 10×10 lattice
- The input samples have two dimensions and are uniformly spaced over an rectangular planar patch
- Synaptic weights are then also two-dimensional vectors
- Images show:

(a) Initial random synaptic weights.

(b) Synaptic weights after about 100 epochs.

(c) Synaptic weights after about 1000 epochs.

(d) Synaptic weights after about 5000 epochs.



Conclusion

- Self-organising neural networks use unsupervised learning
- Unsupervised learning is useful in many practical situations due to missing information about desired network outputs
  - Labelled datasets are prohibitively costly to produce
- Applications include: object classification, pattern recognition, vector quantization, data clustering, feature mapping

References

1. Oja, Erkki. "*Simplified neuron model as a principal component analyzer*." Journal of mathematical biology 15.3 (1982): 267-273.
2. Sanger, Terence D. "*Optimal unsupervised learning in a single-layer linear feedforward neural network.*" Neural networks 2.6 (1989): 459-473.
3. Pearson, Karl. "LIII. On lines and planes of closest fit to systems of points in space." The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 2.11 (1901): 559-572.
4. Hotelling, Harold. "Analysis of a complex of statistical variables into principal components." Journal of educational psychology 24.6 (1933): 417.
5. Kohonen, Teuvo. "Self-organized formation of topologically correct feature maps." Biological cybernetics 43.1 (1982): 59-69.