Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Dipartimento di Fisica e Astronomia
Corso di Laurea in Fisica

Comparison between Oja's and BCM neural networks models in finding useful projections in high-dimensional spaces

Relatore:
Prof. Gastone Castellani

Presentata da:
Tommaso Mariani

Anno Accademico 2016/2017

Ai miei genitori

Abstract

This thesis presents the concept of a neural network starting from its corresponding biological model, paying particular attention to the learning algorithms proposed by Oja and Bienenstock Cooper & Munro. A brief introduction to Data Analysis is then performed, with particular reference to the Principal Components Analysis and Singular Value Decomposition.
The two previously introduced algorithms are then dealt with more thoroughly, going to study in particular their connections with data analysis. Finally, it is proposed to use the Singular Value Decomposition as a method for obtaining stationary points in the BCM algorithm, in the case of linearly dependent inputs.

Sommario

In questa tesi viene presentato il concetto di rete neurale partendo dal suo corrispettivo modello biologico, ponendo particolare attenzione agli algoritmi di apprendimento proposti da Oja e Bienenstock Cooper & Munro. Si esegue poi una breve introduzione alla Data Analysis con particolare riferimento alla tecnica di Analisi delle Componenti Principali e alla Decomposizione ai Valori Singolari.
I due algoritmi introdotti precedentemente vengono poi trattati più approfonditamente, andando a studiare in particolar modo le loro connessioni con l'analisi dati. Viene infine proposto l'utilizzo della Decomposizione ai Valori Singolari come metodo per ricavare i punti stazionari, nell'algoritmo BCM, nel caso di input linearmente dipendenti.

Contents

I Introduction to neural networks and Data Analysis 9

1 Neural networks 11
1.1 Neurons 11
1.1.1 Structure and functional classification 11
1.1.2 Nervous impulse and synapses 12
1.1.3 Synaptic plasticity and learning 13
1.2 Artificial neural networks 14
1.2.1 The perceptron 14
1.2.2 Non-linear perceptron 15
1.3 Learning algorithms 16
1.3.1 The Basic Hebb Rule 17
1.3.2 The BCM Rule 18
1.3.3 Oja's Rule 18

2 Data Analysis 21
2.1 Principal Component Analysis 21
2.1.1 Relation between basis and covariance 22
2.1.2 The eigenvector decomposition 24
2.2 Singular Value Decomposition 25

II Comparison between Oja's and BCM models 27

3 Detailed treatment of neural networks models 29
3.1 Oja's model and PCA 29
3.2 Oja's Rule extension to a network 30
3.3 BCM model 32
3.3.1 Statistical aspects of BCM model 33
3.3.2 Extension to a nonlinear neuron 34
3.4 BCM Rule extension to a network 35

4 Analysis of the fixed points for the BCM rule **37**
4.1 Linearly independent ambient 38
4.1.1 Bidimensional case 38
4.1.2 K-dimensional case 39
4.2 Linearly dependent ambient 39
4.2.1 Bidimensional neuron in a fourth-dimensional ambient 41
4.2.2 Using the SVD to find a base for the kernel and as a pseudoinverse 42

5 Conclusions **43**

Bibliography **45**

Part I

Introduction to neural networks and Data Analysis

Chapter 1

Neural networks

1.1 Neurons

The human nervous system (and animal in general) is formed by nervous tissue, which is in turn formed by cellular units: neurons. These are highly specialized cells dedicated to the transfer, storage and elaboration of information. Tasks that are being carried out thanks to their ability to create an *action potential*, an electric signal that travels long distances along the nerve fiber.
Neurons can also "communicate" with each other via a mechanism called *synapse*, that allows the transmission of the electrical pulse from a cell to another through the use of neurotransmitters (*chemical synapse*) or by a communicating junction (*electrical synapse*).

1.1.1 Structure and functional classification

As seen in figure (1.1), a neuron is composed of four principal parts, having a specific function each:

- the central part of the neuron is called *cellular body*, or *soma*, and this is where the nucleus and others organelles assigned to cellular functions (Golgi apparatus, neurofilaments, endoplasmatic reticulum, etc.) reside.

- the minor fibers which spread from the body are called *dendrites*, their function is to receive electrochemical signals from the adjacent cells and to carry it towards the center of the neuron.

- the terminal part of the cellular body is called *emergence cone* and it is from this zone that the axon and the action potential originate.

- the extension of the cone is called *axon* and it has dimensions that may vary depending on the function of the cell. It is covered in a double layer of protective membranes (neurilemma and myelin sheat) with an electrical insulating function, that interrupts in nodes of Ranvier which facilitate the transmission of the signal.

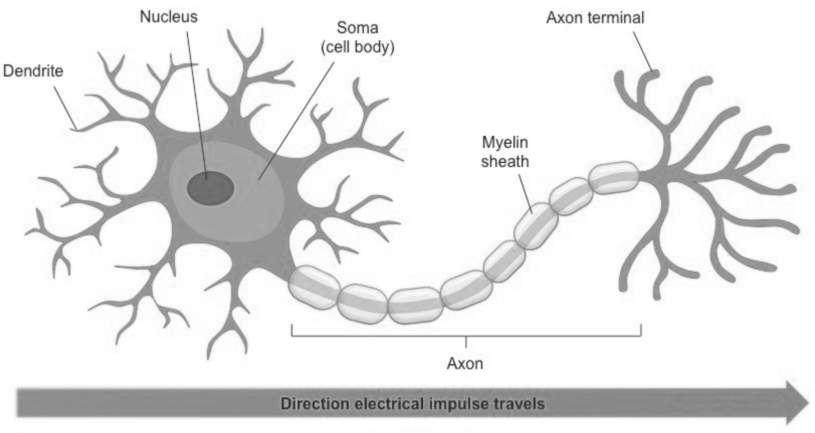


Figure 1.1: Typical structure of a neuron.

Neurons can also be classified in three categories, according to their function and the direction of propagation of the electrical impulse:

- **Afferent neurons (or sensory neurons)**: they acquire stimulus and transmit information from sensorial organs to the central nervous system.
- **Interneurons**: they are in the central nervous system, combining the arriving signals from afferent neurons and transmitting them to efferent neurons.
- **Efferent neurons (or motor neurons)**: they emanate motor type impulses to peripheral organs. They can be distinguished in *somatic motor neurons* that innervate the voluntary striated muscles, and *visceral motor neurons* that innervate the involuntary smooth muscles.

1.1.2 Nervous impulse and synapses

As said before, neurons are capable of transmitting electrical signals along the axon. The signal is generated and propagated through the action potential, a rapid charge variation between the inside and the outside of the cell. Because of the insulating membranes, the nervous information is transmitted between each Ranvier node, in which there is the effective exchange of ions thanks to Na^+ and K^+ channels, that make the passage of the current faster.
Once the signal is generated it propagates, usually unidirectionally, from the dendrites to the axon. From here it goes to the synaptic terminal and is transmitted to the adjacent neuron through the synapse.

Electrical synapse

In the electrical synapse, two different neurons are connected to each other through a communicating junction. These allows the direct passage of electrical current from a cell to the other, so there are no delays in the conduction of the signal.
In general these kind of synapses, unlike the chemical ones, allow the conduction in both directions. It is usually found in nerve connections that regulate the "reflexes", in this case it is necessary indeed a fast transmission between cells.

Chemical synapse

In a chemical synapse the electrical current doesn't pass directly between cells but it is mediated by neurotransmitters. It is formed by three elements: the presynaptic terminal, the synaptic gap and the postsynaptic membrane.
The presynaptic terminal is a highly specialized zone situated at the end of the axon, which contains the neurotransmitters incapsulated in small vesicles (called *synaptic vesicles*) that are released, upon the arrival of the action potential, in the synaptic gap.
Here neurotransmitters enter in contact with the post-synaptic membrane and, after being absorbed by specific neural receptors, generate another action potential in the new cell.

1.1.3 Synaptic plasticity and learning

By the term "synaptic plasticity" is meant the ability of the nervous system to change the strength of interneural relations (synapses) during time, in response to increasing or decreasing activity. It is also believed that neurons are able to create new connections and eliminate some.
This property allows the nervous system to modify its structure and hence its functionality in a more or less durable way, so the analogy with the concept of "experience" is obvious.

Hebb's Principle

Hebb's principle, proposed by scientist Donald Hebb in 1949, is an attempt to describe the process of modifying synaptic weights as a consequence of the stimulation, by the presynaptic cell, of the postsynaptic cell. Hebb stated that:

> *"When an axon of neuron A is close enough to neuron B to activate it repeatedly and persistently, some growth or metabolic change in one or both cells occurs, causing an increase in neuron A's effectiveness in exciting neuron B".*

The models that follow this principle are called *Hebbian learning models.* However, this rule alone does not fully explain the model of synaptic plasticity, but rather provides a wrong and unstable one, as synaptic weights can only increase over time.

Stent's Principle

The Stent principle, proposed by scientist Gunther S. Stent in 1973, is an attempt to correct the neural learning model proposed by Hebb introducing a process that explains the decreasing of synaptic weights. Stent said that:

> *"When the attempt by a presynaptic neuron A to activate a postsynaptic neuron B cell fails repeatedly and persistently, some process of decreasing or metabolic change occurs in one or both cells, causing a decrease in neuron A's effectiveness in exciting neuron B".*

The two principles are therefore in competition with each other and, together, are responsible for synaptic plasticity. This process is in fact also known as "spatial competition between neurons".

1.2 Artificial neural networks

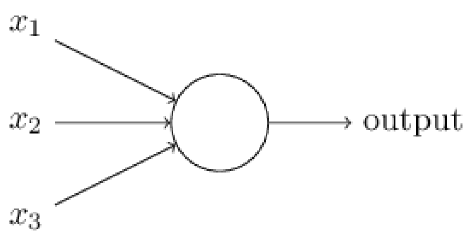
An artificial neural network is a mathematical model, made up of artificial neurons, inspired by the neural network's biological model where neurons are densely connected to each other. These artificial neurons are really simpler models, they receive several input and transmit just a single output, it's in fact the non-linearity of interconnections the cause of the complexity of the system.

1.2.1 The perceptron

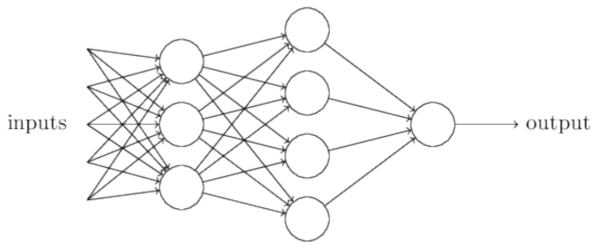
The simplest model for a neuron goes under the name of *perceptron.* The perceptron takes some variables $(x_1, x_2, \dots, x_n)$, one for each synapse, as input and returns only one binary output y. We also introduce the *synaptic weights* $(w_1, w_2, \dots, w_n)$, that express the importance of inputs on the output.
The neuronal output (0 or 1) will then be determined by the fact that the weighted sum of the inputs $\sum_j w_j x_j$ is greater than or equal to a *threshold value* indicated with θ. Written in algebraic form:

$$y = \begin{cases} 0 & \sum_j w_j x_j \leq \theta \\ 1 & \sum_j w_j x_j > \theta \end{cases} \tag{1.1}$$

By joining together some of these fundamental units, the artificial neural network is formed, as shown in figure(1.2).



(a) perceptron



(b) artificial neural network

Figure 1.2: Structure of a perceptron and of an artificial neural network

Also from the figure it is noted that the perceptrons are divided into subgroups called network *layers*. Although the first of these simply does the weighted average of the inputs, the second performs the weighted average on the results of the first layer, thus giving rise to levels of deeper complexity and abstractness than the previous ones. A multi-layered perceptron is therefore able to perform complex and highly sophisticated calculations of a single-layered one.

1.2.2 Non-linear perceptron

To allow a perceptrons network to learn how to solve a real problem, it needs to be able to independently modify the weights and threshold values. This ability is achieved through what is called the *learning algorithm*. However, in the perceptron model defined above, a small variation of a single weight can result in a large variation in output.
To solve this problem a small change of the artificial neuron, called *sigmoid neuron* (or nonlinear) is introduced. Just like normal perceptron, the nonlinear neuron receives the inputs $(x_1, x_2, \ldots, x_n)$ and, also in complete analogy, has a θ threshold value, but the output will not be only 0 or just 1. Indeed, neuronal output now follows the *sigmoid function* given by:

$$\sigma(z) \equiv \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-\sum_j w_j x_j - \theta}} \tag{1.2}$$

whose trend is shown in figure (1.3).

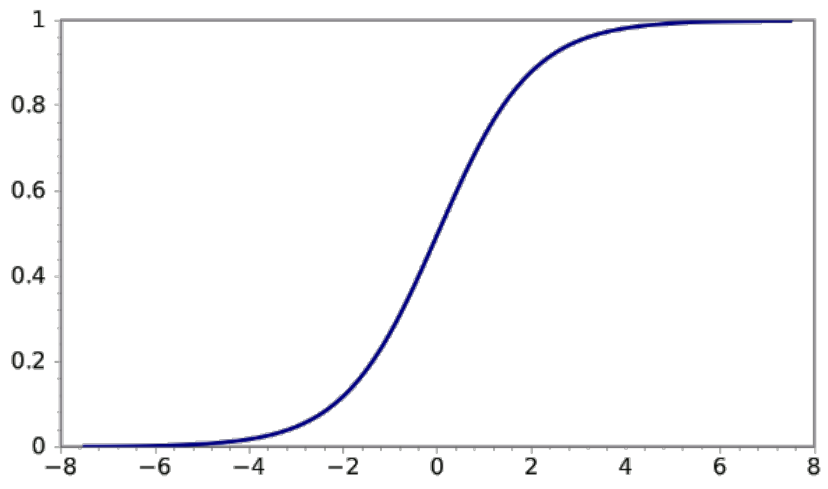


Figure 1.3: Trend of the sigmoid function

We note that the function is a smoother version of the step function (typical of the perceptron). It is for this reason that for small variations of Δw_j and $\Delta\theta$ small variations Δy are produced.

1.3 Learning algorithms

The study of synaptic plasticity consists in analyzing how synapses are modified, as a consequence of their activation during what is called the learning period: a time frame during which a set of predetermined inputs and outputs is presented and the network response is then evaluated.
The way neurons learn can be divided into three categories:

- in *unsupervised learning* a neural network responds to a set of inputs during a training period only based on its intrinsic connections and neural dynamics.
  The network then self-organizes according to the chosen learning rule and the type of the inputs presented.

- in *supervised learning* instead a set of relationships between input and output from the outside is imposed. The network then changes its synapses until the same pattern emerges as a result of the learning process.

- *reinforcement learning* is an intermediate case between the previous ones. Neuronal output is not imposed from the outside but there is a feedback rule that alters synaptic weights based on the network performance over time.

The synaptic change rules assume the form of differential equations describing the change of synaptic weights according to the presynaptic activity, postsynaptic activity and other possible factors.
In the various rules we will study, neuronal activity is described by a continuous variable. Presynaptic activity is nothing but the input provided and will be denoted, in vector notation, with the column vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ while synaptic weights are indicated by the column vector $\mathbf{w} = (w_1, w_2, \ldots, w_n)$. The neuronal output therefore assumes the form of the scalar product:

$$y = \sum_j w_j x_j = \mathbf{w} \cdot \mathbf{x} \tag{1.3}$$

1.3.1 The Basic Hebb Rule

The simplest synaptic plasticity rule that follows the Hebb's principle takes the form:

$$\tau_w \frac{d\mathbf{w}}{dt} = y\mathbf{x} \tag{1.4}$$

where τ_w is a time constant that controls the rate at which the weights change, it's reciprocal is called *learning rate.*
This equation, which is called the basic Hebb rule, implies that if to a determinated presynaptic activity corresponds an high postsynaptic activity then the synaptic weight grows. The direct way to compute the weights changes, induced by a series of inputs, is to sum the small changes caused by each of them separately.
A convenient alternative is to average over all of the different inputs and compute weights change induced by this average. As long as the synaptic weights change slowly enough, the averaging method provides a good approximation of the weight changes produced by the set of inputs. The Hebb rule of (1.4) becomes

$$\tau_w \frac{d\mathbf{w}}{dt} = \langle y\mathbf{x} \rangle \tag{1.5}$$

In unsupervised learning, y is determined by (1.3) and then we can rewrite the averaged plasticity rule as

$$\tau_w \frac{d\mathbf{w}}{dt} = \langle \mathbf{w} \cdot \mathbf{x} \cdot \mathbf{x} \rangle = \mathbf{w} \cdot \mathbf{C_x} \tag{1.6}$$

where $\mathbf{C_x}$ is the input correlation matrix given by $\mathbf{C_x} = \langle \mathbf{x} \cdot \mathbf{x} \rangle$.
Whether or not the pre and postsynaptic activity variables are restricted to nonnegative values, the basic Hebb rule is unstable: the length of the weight vector grows continuously and diverges.

Sometimes, synaptic modification is modeled as a discrete rather than continuous process, particularly if the learning procedure involves the sequential presentation of inputs. In this case, equation (1.6) is replaced by a discrete updating rule:

$$\mathbf{w} \rightarrow \mathbf{w} + \epsilon \mathbf{C}_{\mathbf{x}} \cdot \mathbf{w} \tag{1.7}$$

where ϵ is a parameter analogous to the learning rate $1/\tau_w$.

1.3.2 The BCM Rule

In 1982 Bienestock, Cooper and Munro suggested an alternative plasticity rule that introduces a new mechanism

$$\tau_w \frac{d\mathbf{w}}{dt} = y\mathbf{x}(y - \theta) \tag{1.8}$$

where θ acts as a threshold on the postsynaptic activity that determines whether the synapses are strengthened or weakened.
If the threshold θ is held fixed, the BCM rule, like the basic Hebb rule, is unstable. Synaptic modification can be stabilized then by allowing the threshold to vary. The critical condition for stability is that θ must grow more rapidly than y as the output activity grows large. The sliding threshold is then determined by the equation

$$\tau_\theta \frac{d\theta}{dt} = y^2 - \theta \tag{1.9}$$

where τ_θ sets the time for modification of the threshold. This is usually slower than the presentation of the individual presynaptic inputs, but faster than the rate at which the weights change, which is determined by τ_w.

1.3.3 Oja's Rule

The BCM rule stabilizes Hebbian plasticity by means of a sliding threshold that reduces synaptic weights if the postsynaptic neuron becomes too active. A more direct weight to stabilize a Hebbian rule is to add terms that depend explicitly on the weights. This leads to some form of weight normalization, which corresponds to the idea that postsynaptic neurons can support only a fixed total synaptic weight.

A constraint on the sum of the squares of the synaptic weights can be imposed by using a modification of the basic Hebb rule known as the Oja's rule

$$\tau_w \frac{d\mathbf{w}}{dt} = y\mathbf{x} - \alpha y^2 \mathbf{w} \tag{1.10}$$

where α is a positive constant. The stability can be established by taking the dot product of the equation (1.10) with the vector $\mathbf{w}$ to obtain

$$\tau_w \frac{d|\mathbf{w}|^2}{dt} = 2y^2(1 - \alpha|\mathbf{w}|^2) \tag{1.11}$$

This indicates that $|\mathbf{w}|^2$ will relax over time to the value $1/\alpha$, which prevents the weights from growing without bound.

Chapter 2

Data Analysis

From the previous chapter, it is clear that the high utility of neural networks lies in their flexibility due to their ability to learn, in fact these networks can be used to simulate any function using only data observations. This is particularly useful in data analyzation where the complexity or size of these is very high, and therefore processing is too difficult through normal procedures.
In statistical analysis, data that enclose a high number of features are called *high-dimensional data*, when analyzing this type of data it's easy to get into what goes under the name of *curse of dimensionality*. What happens when working with many dimensions is that space volume increases so rapidly that data becomes "scarce": so there is a need for data that increases exponentially with dimensions to achieve significant static results.

To solve this problem, is used a statistical technique called *projection pursuit*: this method allows the analysis of the "most interesting" data distributions, projected on appropriate subspaces. The underlying idea is to find the representations that retain most of the data structure information but which can be analyzed using traditional techniques. As each projection is found, the data are reduced by removing the component along that projection, and the process is repeated to find new ones. Often, projections which deviate more from a normal distribution are considered to be more interesting due to the central limit theorem.

2.1 Principal Component Analysis

Principal component analysis (PCA) is a multivariate technique that analyzes data with the goal to extract important information and to represent it as a new set of orthogonal variables called Principal Components.
Mathematically, Principal Component Analysis is defined as an orthogonal linear transformation that transports the data to a new coordinate system, such that the greatest

variance by some projected distribution of the data comes to lie on the first coordinate (called the first Principal Component), the second greatest variance on the second coordinate and so on.

2.1.1 Relation between basis and covariance

Consider a data matrix $\mathbf{X}$, composed of all the possible inputs vectors, with row zero mean (this could be done by simply shifting all the values, without loss in generality), where each of the m rows represents a particular kind of measure, and each of the n columns is a different sample of our data set.

$$\mathbf{X} = \left[\begin{pmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{m1} \end{pmatrix} \begin{pmatrix} x_{21} \\ x_{22} \\ \vdots \\ x_{m2} \end{pmatrix} \cdots \begin{pmatrix} x_{1n} \\ x_{2n} \\ \vdots \\ x_{mn} \end{pmatrix} \right]$$

We now want to seek a linear transformation matrix $\mathbf{T}$, to re-express the original data set into the new bases $\mathbf{Y}$:

$$\mathbf{Y} = \mathbf{TX} \tag{2.1}$$

where we note that $\mathbf{T}$ must be $m \times m$.
Geometrically the expression (2.1) can have many interpretations:

- $\mathbf{T}$ is a rotation and a stretch that transform $\mathbf{X}$ into $\mathbf{Y}$
- the rows of $\mathbf{T}$, $\{\mathbf{e}_1, \dots, \mathbf{e}_m\}$, are a set of new basis vectors for expressing the columns of $\mathbf{X}$.

The best basis to re-express the data contained in $\mathbf{X}$, is the one that has as directions the ones with largest variances in our measurement space, as these contain the dynamic of interest.

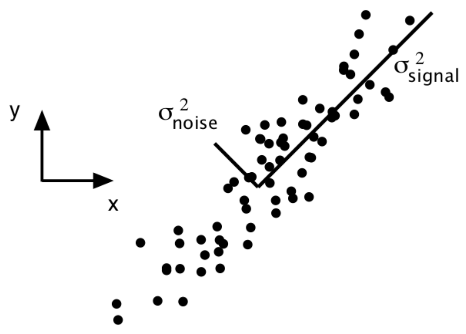


Figure 2.1: Example of data acquisition with low noise.

In figure (2.1) an example of a possible data acquisition with low noise is represented. We can define the *signal-to-noise ratio (SNR)* as:

$$SNR = \frac{\sigma^2_{signal}}{\sigma^2_{noise}}$$

A high SNR ($\gg 1$) indicates a high precision measurement, while low SNR indicates very noisy data. Obviously we want to isolate the signal from the noise and consider only the first one.

To do so we need first to define the *covariance*. Consider two sets of measurements with zero means

$$A = \{a_1, a_2, \ldots, a_n\} \qquad\qquad B = \{b_1, b_2, \ldots, b_n\}$$

the variance of A and B are then defined as

$$\sigma^2_A = \frac{1}{n}\sum_i a_i^2 \qquad\qquad \sigma^2_B = \frac{1}{n}\sum_i b_i^2$$

The *covariance* between A and B is a generalization of the latters

$$\sigma^2_{AB} = \frac{1}{n}\sum_i a_i b_i$$

The covariance then measures the degree of linear relationship between two variables, in particular:

- σ_{AB} is zero in and only if A and B are uncorrelated.
- $\sigma_{AB} = \sigma_A = \sigma_B$ if and only if $A = B$.

To generalize this concept in higher dimensional spaces we may note that the sets A and B can be expressed through the vector notation

$$\mathbf{a} = (a_1, a_2, \ldots, a_n)$$
$$\mathbf{b} = (b_1, b_2, \ldots, b_n)$$

the covariance is then

$$\sigma^2_{\mathbf{ab}} = \frac{1}{n}\mathbf{ab}^T$$

Let us consider the $m \times n$ matrix $\mathbf{X}$ defined above, each row then corresponds to all measurements of a particular type (for example vector $\mathbf{a}$ or $\mathbf{b}$) and each column corresponds to a repetition of the measure:

$$\mathbf{X} = \begin{bmatrix} (a_1, a_2, \ldots, a_n) \\ (b_1, b_2, \ldots, b_n) \\ \vdots \\ (m_1, m_2, \ldots, m_n) \end{bmatrix}$$

Now we can define the *covariance matrix* $\mathbf{C_X}$:

$$\mathbf{C_X} = \frac{1}{n}\mathbf{X}\mathbf{X}^T$$

that is obviously $m \times m$. It may be noted that the diagonal terms are the variance of particular measurements tipes, the off-diagonal terms instead are the covariance between measurement types.

The covariance matrix $\mathbf{C_Y}$ (the covariance matrix $\mathbf{C_X}$ expressed in the new basis) should now be optimized in a precise way:

- all off-diagonal terms in $\mathbf{C_Y}$ should be zero. Thus, $\mathbf{C_Y}$ must be a diagonal matrix. In this way every different type of measurement is meant to be uncorrelated with the others.
- each successive dimension in $\mathbf{Y}$ should be rank-ordered according to variance.

2.1.2 The eigenvector decomposition

The easiest way to diagonalize $\mathbf{C_Y}$ is to assume the base $\{\mathbf{e}_1, \mathbf{e}_2 \ldots, \mathbf{e_m}\}$ to be orthonormal (so $\mathbf{T}$ is orthonormal) and using the eigenvector decomposition.
We need to find some orthonormal matrix $\mathbf{T}$ in $\mathbf{Y} = \mathbf{TX}$ such that $\mathbf{C_Y} = \frac{1}{n}\mathbf{Y}\mathbf{Y}^T$ is a diagonal matrix. The rows of $\mathbf{T}$ are then the *principal components* of $\mathbf{X}$.
We begin by rewriting $\mathbf{C_Y}$ in terms of $\mathbf{T}$:

$$\begin{aligned} \mathbf{C_Y} &= \frac{1}{n}\mathbf{Y}\mathbf{Y}^T = \frac{1}{n}(\mathbf{TX})(\mathbf{TX})^T = \\ &= \frac{1}{n}\mathbf{T}\mathbf{X}\mathbf{X}^T\mathbf{T}^T = \mathbf{T}(\frac{1}{n}\mathbf{X}\mathbf{X}^T)\mathbf{T}^T = \mathbf{T}\mathbf{C_X}\mathbf{T}^T \end{aligned} \tag{2.2}$$

By the spectral theorem of linear algebra it is known that any symmetric matrix $\mathbf{A}$ is diagonalized by an orthogonal matrix of its eigenvectors, $\mathbf{A}$=$\mathbf{EDE}^T$, where $\mathbf{D}$ is a diagonal matrix and $\mathbf{E}$ is a matrix of eigenvectors of $\mathbf{A}$ arranged as columns.
If the matrix $\mathbf{T}$ is a matrix where each row $\mathbf{e}_i$ is an eigenvector of $\mathbf{C_X}$ the equation (2.2) can be rewritten as follows

$$\begin{aligned} \mathbf{C_Y} &= \mathbf{T}\mathbf{C_X}\mathbf{T}^T = \mathbf{T}(\mathbf{EDE}^T)\mathbf{T}^T = \\ &= \mathbf{T}(\mathbf{T}^T\mathbf{DT})\mathbf{T}^T = (\mathbf{TT}^T)\mathbf{D}(\mathbf{TT}^T) = \mathbf{D} \end{aligned} \tag{2.3}$$

beacuse we have selected $\mathbf{A} = \mathbf{C_X}$, then $\mathbf{T}^T = \mathbf{E}$. Note also that $\mathbf{T}^T = \mathbf{T}^{-1}$.

It is now evident that the choice of $\mathbf{T}$ diagonalizes $\mathbf{C_Y}$, that is exactly the goal of PCA. To summarize:

- the principal components of $\mathbf{X}$ are the eigenvectors of $\mathbf{C_X} = \frac{1}{n}\mathbf{X}\mathbf{X}^T$, which in turn are the rows of the transformation matrix $\mathbf{T}$.
- the i^{th} diagonal value of $\mathbf{C_X}$ is the variance of $\mathbf{X}$ along $\mathbf{e_i}$.

2.2 Singular Value Decomposition

An useful procedure used in Principal Component Analysis is the Singular Value Decomposition of a matrix, a technique that generalizes the concept of eigenvalues, eigenvectors and inverse of a non-square matrix.

Let $\mathbf{X}$ be a $m \times n$ matrix as described in the previous section and $\mathbf{X}\mathbf{X}^T$ be a square, symmetric $m \times m$ matrix. Then the factorization exists:

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$$

where $\mathbf{U}$ is a $m \times m$ orthogonal matrix, Σ is a diagonal rectangular $m \times n$ matrix and $\mathbf{V}^T$ is the transpose of a $n \times n$ orthogonal matrix.
The diagonal elements of Σ are called *singular values* and they have the following properties

$$\sigma_i \geq 0 \qquad\qquad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$$

where r is the rank of $\mathbf{X}$.
The SVD is so important because the columns of $\mathbf{V}$ (called right-singular vectors) are the eigenvectors of $\mathbf{X}^T\mathbf{X}$, and the columns of $\mathbf{U}$ (called left-singular vectors) are the eigenvectors of $\mathbf{X}\mathbf{X}^T$. The covariance matrix $\mathbf{C_X}$ can be written (except for a normalization constant) as

$$\mathbf{C_X} = \mathbf{X}\mathbf{X}^T = \mathbf{U}\Sigma\mathbf{V}^T\mathbf{V}\Sigma^T\mathbf{U}^T = \mathbf{U}\Sigma^2\mathbf{U}^T \tag{2.4}$$

from which we deduce that the eigenvectors of the covariance matrix $\mathbf{C_X}$ are exactly the left-singular vectors of $\mathbf{U}$, and the singular values σ_i are the square root of the λ_i, eigenvalues of the covariance matrix.
Since there are algorithms that can calculate the singular values of a matrix without evaluating the $\mathbf{C_X}$, the SVD is a very efficient method to obtain the principal components from a data matrix $\mathbf{X}$.

SVD as a pseudo-inverse

The importance of SVD also lies in considering the decomposition as a generalization of the concept of inverse matrix. In fact, if a matrix $m \times n$ has a singular value decomposition $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$ then its pseudoinverse is

$$\mathbf{X}^{-1} = \mathbf{V}\Sigma^{-1}\mathbf{U}^T \tag{2.5}$$

where Σ^{-1} is the pseudoinverse of Σ, which is formed by replacing every non-zero diagonal entry by its reciprocal and transposing the resulting matrix.

The importance of polynomial moments

Although many models of synaptic plasticity are based on second order statistics, for example the extraction of the principal components, not always second order polynomial moments are sufficent to classify the important aspects of a distribution. Consequently, polynomials of high order must be used, paying attention to their sensibility to the outliers.
The BCM model, for example, uses moments of greater order and is therefore a great candidate as learning algorithm for projection pursuit.

Part II

Comparison between Oja's and BCM models

Chapter 3

Detailed treatment of neural networks models

Oja's learning rule, or simply Oja's model, named after scientist Erkki Oja, is a model of how neurons in artificial neural networks change connection strength, and learn, over time. It is a modification of the Basic Hebb's Rule (1.4) that, through weights normalization, solves the stability problems and generates an algorithm for principal components analysis.

BCM model, named after scientists Bienenstock Cooper and Munro, unlike Oja, is based on biological considerations, in particular how the connections of the visual cortex change according to the inputs presented. We will find however that this is also a useful model for statistical analysis.

3.1 Oja's model and PCA

Oja's rule is closely related to the Principal Component Analysis, in fact if an artificial neural network uses this learning algorithm, the weight vector then converges, after a while, to the first eigenvector of the covariance matrix. To prove this we need to express the rule (1.10) in a more convenient way.
As explained in the first chapter, Oja's rule for synaptic plasicity is (combining the two learning constants in a single one for simplicity):

$$\frac{d\mathbf{w}}{dt} = \alpha(y\mathbf{x} - y^2\mathbf{w}) \tag{3.1}$$

Now this can be expressed through the vector notation, where $y = \mathbf{x} \cdot \mathbf{w} = \mathbf{x}^T\mathbf{w} = \mathbf{w}^T\mathbf{x}$ as we are using *column* vectors:

$$\frac{d\mathbf{w}}{dt} = \alpha(\mathbf{x}\mathbf{x}^T\mathbf{w} - \mathbf{w}^T\mathbf{x}\mathbf{x}^T\mathbf{w}\mathbf{w}) \tag{3.2}$$

The condition for the convergence of the equation to a stable point is $\frac{d\mathbf{w}}{dt} = 0$, or the non-variation of synaptic weights. Furthermore, when the learning algorithm is run for a long time with a continuously change of inputs, one can look at the average behaviour because it is assumed that the time for a synaptic variation is faster than the analysis time. If it is also assumed that the inputs have row zero mean (without loss in generality) then the former equation becomes:

$$\mathbf{Cw} - \mathbf{w}^T\mathbf{Cww} = 0 \tag{3.3}$$

where $\mathbf{C} = E[\mathbf{xx}^T]$ is the covariance matrix. It must be noted that the quadratic form $\mathbf{w}^T\mathbf{Cw}$ is a scalar, which will be called λ for an obvious reason, the new equation therefore is:

$$\mathbf{Cw} = \lambda\mathbf{w} \tag{3.4}$$

that clearly is the eigenvalue-eigenvector equation for the covariance matrix $\mathbf{C}$.

This shows that if the weights converge in the Oja learning rule, then the weight vector becomes one of the eigenvectors of the input covariance matrix, and the output of the neuron becomes the corresponding Principal Component, as Principal Components are defined as the inner products between the eigenvectors and the input vectors. Then the simple neuron following Oja's model becomes a Principal Component analyzer.
However what is found in this case is only the first Principal Component, for a more in-depth analysis we need to extend the algorithm for a single neuron to a complete network.

3.2 Oja's Rule extension to a network

The most used algorithm, that extends Oja's rule to a lateral connections network, was developed by scientist Terence Sanger who called it the Generalized Hebbian Algorithm. This algorithm combines Oja's rule and the Gram-Schmidt orthogonalization process to realize the synaptic weights modification rule:

$$\frac{\partial w_{ij}}{\partial t} = \alpha y_i (x_j - \sum_{k=1}^{i} w_{kj} y_k) \tag{3.5}$$

where w_{ij} represents connection strength between the j-th input and the i-th output (see figure 3.1 for clarification).
We note the term $(-\alpha y_i \sum_k w_{kj} y_k)$, which represents a decay term, is the weighted sum of the first i-th neuronal activations.

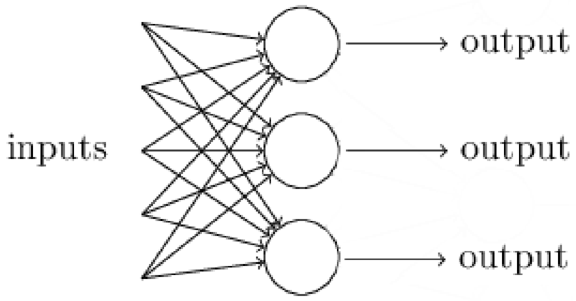


Figure 3.1: Example of the structure of a single-layer artificial neural networks with several outputs.

The Generalized Hebbian Algorithm can be seen as the application of Oja's rule for a single neuron, in fact, by rewriting the former equation as:

$$\frac{\partial w_{ij}}{\partial t} = \alpha[(y_i x_j - y_i \sum_{k=1}^{i-1} w_{kj} y_k) - y_i^2 w_{ij}]$$

and introducing the modified input

$$x_j^{(i)} = x_j - \sum_{k=1}^{i-1} w_{kj} y_k \qquad (3.6)$$

the equation (3.5) can now be written in the same form of (3.1):

$$\frac{\partial w_{ij}}{\partial t} = \alpha(y_i x_j^{(i)} - y_i^2 w_{ij}) \qquad (3.7)$$

The input driving y_i is modified by subtracting all the i-th exctracted components. Such a technique is called *deflation.*

From these equations it's easy to see that the first neuron is using the simple Oja's rule:

$$\frac{\partial w_{1j}}{\partial t} = \alpha(y_1 x_j - y_1^2 w_{1j})$$

but the second one uses a slightly different rule:

$$\frac{\partial w_{2j}}{\partial t} = \alpha[(y_2 x_j - y_2 y_1 w_{1j}) - y_2^2 w_{2j}]$$

and so on. From these considerations we may note that this algorithm in *non-local*, or the i-th neuron, to properly perform the Principal Component Analysis, needs to know the w_{ij} synaptic weights of neurons before him.
Extended in this way the algorithm, the weights vectors of each of the neurons converge to a different Principal Component direction. We can therefore have a Principal Component for each neuron that forms the network.

3.3 BCM model

BCM model is an unsupervised learning algorithm that uses a sliding threshold to solve the problem of the instability of the Basic Hebbian Rule.

In the original article Bienenstock Cooper and Munro proposed that, for a generic learning algorithm, the modifyng rule is:

$$\frac{d\mathbf{w}}{dt} = \alpha[\phi(y, \theta) \cdot \mathbf{x} - \epsilon \mathbf{w}] \tag{3.8}$$

where $\phi(y)$ is a scalar function of the postsynaptic activity y that changes sign when this reaches the threshold value θ. The $-\epsilon\mathbf{w}$ term produces a uniform decrease for all synaptic weights but ϵ is usally very small and can be neglected.
We then have two possible behaviours of the synaptic rule:

- $\frac{dw_j}{dt} > 0$ (increased synaptic efficiency), when $x_j > 0$ and $y > \theta$
- $\frac{dw_j}{dt} < 0$ (decreased synaptic efficiency), when $x_j > 0$ and $y < \theta$

For their model they chose $\theta = E[y^2]$ and a quadratic form for the ϕ function:

$$\phi(y, \theta) = y(y - \theta)$$

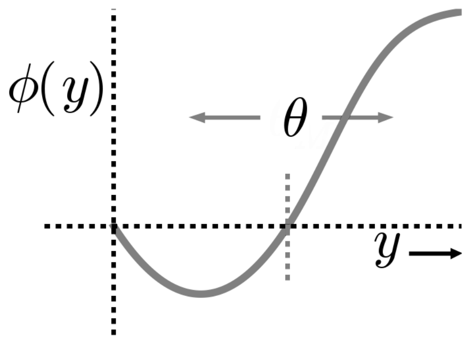


Figure 3.2: Trend of the ϕ function. It's important to note that θ is a sliding threshold that depends on output over time.

Then, as explained in chapter one, the BCM rule for synaptic plasticity becomes simply:

$$\frac{d\mathbf{w}}{dt} = \alpha y(y - \theta)\mathbf{x} \tag{3.9}$$

The main objective of the BCM model, derived from biological considerations, is to maximize the index of selectivity defined as:

$$s(\mathbf{w}) = 1 - \frac{E[\mathbf{x}^T\mathbf{w}]}{\max[\mathbf{x}^T\mathbf{w}]} \tag{3.10}$$

which describes the reactivity of a neuron for various inputs. This is clearly maximized when the neuron responds only to a single input (or a few), like the **w** vector is orthogonal to almost **x** input vectors.

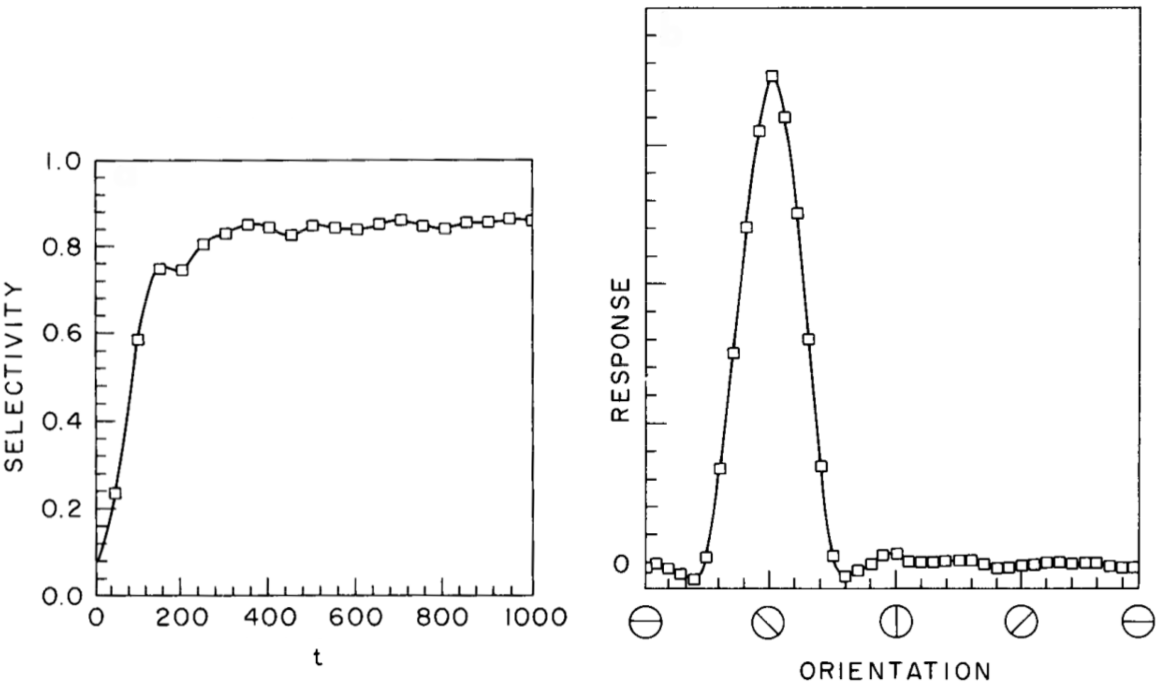


Figure 3.3: The original image about neuronal selectivity of the article by Bienestock Cooper and Munro. The right image shows how a neuron develops a certain preference among a different types of inputs (in this case the orientation of a light rod). The left image instead shows the increasing of neural selectivity index $s(\mathbf{w})$ over time, as the neuron begins to develop preference.

3.3.1 Statistical aspects of BCM model

The BCM model's theory can also be formulated through the definition of an objective function to minimize. This allows the interpretation of the model behaviour from a statistical point of view by studying the polynomial moments of the distribuition.

We define the threshold as before $\theta = E[y^2]$ and two functions $\hat{\phi}(y, \theta) = y^2 - \frac{1}{2}y\theta$ and $\phi(y, \theta) = y(y - \theta)$ for simplicity of notation.

The purpose is now to find directions for which the projected distribution is far from Gaussian; for computational efficiency we have to use low degree polynomial moments, but at least of third degree as second ones do not give any information on multimodality. Then the loss function can be defined as:

$$\begin{aligned} L_{\mathbf{w}}(\mathbf{x}) &= -\alpha \int_0^{(\mathbf{x}\cdot\mathbf{w})} \hat{\phi}(s,\theta) ds = \\ &= -\alpha \left\{ \frac{1}{3}(\mathbf{x}\cdot\mathbf{w})^3 - \frac{1}{4} E[(\mathbf{x}\cdot\mathbf{w})^2](\mathbf{x}\cdot\mathbf{w})^2 \right\} \end{aligned} \tag{3.11}$$

as we recall that $y = \mathbf{x}\cdot\mathbf{w}$. The risk function then is (the expectation value of the loss):

$$\begin{aligned} R_{\mathbf{w}} &= -\alpha E\left[\frac{1}{3}(\mathbf{x}\cdot\mathbf{w})^3 - \frac{1}{4} E[(\mathbf{x}\cdot\mathbf{w})^2](\mathbf{x}\cdot\mathbf{w})^2 \right] = \\ &= -\alpha \left(\frac{1}{3} E[(\mathbf{x}\cdot\mathbf{w})^3] - \frac{1}{4} E^2[(\mathbf{x}\cdot\mathbf{w})^2] \right) \end{aligned} \tag{3.12}$$

Since the risk is continuously differentiable, its minimization can be achieved via the gradient descent method with respect to $\mathbf{w}$:

$$\begin{aligned} \frac{dw_i}{dt} &= -\frac{\partial R_{\mathbf{w}}}{\partial w_i} = \alpha\{E[(\mathbf{x}\cdot\mathbf{w})^2 x_i] - E[(\mathbf{x}\cdot\mathbf{w})^2]E[(\mathbf{x}\cdot\mathbf{w})x_i]\} = \\ &= \alpha E[\phi(\mathbf{x}\cdot\mathbf{w},\theta)x_i] \end{aligned} \tag{3.13}$$

that is exactly the equation (3.9). The weight vector will then converge, following this rule, to the direction in which the data follow a distribution that looks more like gaussian. Once projected along this direction, data then have a smaller dimension and a higher probability to differ from a normal distribution.

3.3.2 Extension to a nonlinear neuron

This learning algorithm can be improved by making the risk function, and then the BCM rule, insensitive to eventual outliers. To do so we consider a nonlinear neuron in which the neuron's activity is defined to be $y = \sigma(\mathbf{x}\cdot\mathbf{w})$, where σ is the already known sigmoid function.
For the nonlinear neuron, the only difference is that the threshold is defined as $\theta = E[\sigma^2(\mathbf{x}\cdot\mathbf{w})]$. The loss function is then given by:

$$\begin{aligned} L_{\mathbf{w}}(\mathbf{x}) &= -\alpha \int_0^{\sigma(\mathbf{x}\cdot\mathbf{w})} \hat{\phi}(s,\theta) ds = \\ &= -\alpha \left\{ \frac{1}{3}\sigma^3(\mathbf{x}\cdot\mathbf{w}) - \frac{1}{4} E[\sigma^2(\mathbf{x}\cdot\mathbf{w})]\sigma^2(\mathbf{x}\cdot\mathbf{w}) \right\} \end{aligned} \tag{3.14}$$

and the gradient of the risk is then:

$$\begin{aligned}\nabla_{\mathbf{w}} R_{\mathbf{w}} &= -\alpha E[\sigma^2(\mathbf{x}\cdot\mathbf{w})\sigma'\mathbf{x}] - E[\sigma^2(\mathbf{x}\cdot\mathbf{w})]E[\sigma(\mathbf{x}\cdot\mathbf{w})\sigma'\mathbf{x}] = \\ &= \alpha E[\phi(\sigma(\mathbf{x}\cdot\mathbf{w}),\theta)\sigma'\mathbf{x}]\end{aligned} \tag{3.15}$$

where σ' represents the derivative of σ at the point $(\mathbf{x}\cdot\mathbf{w})$. It is the multiplication by σ' that reduces the sensitivity to outliers in the differential equation, since for outliers σ' is close to zero.

Howewer what is found in this case is only one of the possible directions, as for Oja's one we now need to extend the algorithm for a single neuron to a complete network.

3.4 BCM Rule extension to a network

We now define a network where the activity of a single neuron is inhibited by the others (the so called Feed-Forward Inhibition). The activity of neuron k in the network is $y_k = \mathbf{x}\cdot\mathbf{w}_k$, where $\mathbf{m}_k$ is the synaptic weight vector of the neuron. The inhibited activity and threshold of that neuron is defined as:

$$\tilde{y}_k = y_k - \alpha\sum_{j\neq k} y_j \qquad\qquad \tilde{\theta}_k = E[\tilde{y}_k^2]$$

For the feed-forward network the loss function is similar to the one defined in a single feature extraction, with the exception that the activity y is now replaced by $\tilde{y}$. Therefore the risk for neuron k is given by:

$$R_k = -\alpha\left\{\frac{1}{3}E[\tilde{c}_k^3] - \frac{1}{4}E^2[\tilde{c}_k^2]\right\} \tag{3.16}$$

and then the total risk is given by

$$R = \sum_{k=1}^{N} R_k$$

To find the gradient of R it is better to express some terms first as:

$$\begin{aligned}&\frac{\partial\tilde{y}_k}{\partial w_j} = -\alpha\mathbf{x} \qquad \frac{\partial\tilde{y}_k}{\partial w_k} = \mathbf{x}\\ &\frac{\partial R_j}{\partial w_k} = \frac{\partial R_j}{\partial\tilde{y}_j}\frac{\partial\tilde{y}_j}{\partial w_k} = -\alpha\frac{\partial R_j}{\partial w_j}\\ &\frac{\partial R_k}{\partial w_k} = \frac{\partial R_k}{\partial\tilde{y}_k}\frac{\partial\tilde{y}_k}{\partial w_k} = -\alpha\{E[\tilde{y}_k^2\mathbf{x}] - E[\tilde{y}_k^2]E[\tilde{y}_k\mathbf{x}]\}\end{aligned}$$

the total gradient then becomes:

$$\begin{aligned} \frac{\partial R}{\partial w_k} &= \frac{\partial R_k}{\partial w_k} - \alpha \sum_{j\neq k} \frac{\partial R_j}{\partial w_j} = \\ &= \alpha \left\{ E[\phi(\tilde{y}_k, \theta_k)\mathbf{x}] - \alpha \sum_{j\neq k} E[\phi(\tilde{y}_j, \theta_j)\mathbf{x}] \right\} \end{aligned} \tag{3.17}$$

When the nonlinearity of the network is included, the inhibited activity is defined as $\tilde{y} = \sigma(y_k - \alpha \sum_{j\neq k} y_j)$. θ_k and R_k are defined as before, however in this case we have:

$$\frac{\partial \tilde{y}_k}{\partial w_j} = -\alpha\sigma'(\tilde{y})\mathbf{x} \qquad\qquad \frac{\partial \tilde{y}_k}{\partial w_k} = \sigma'(\tilde{y})\mathbf{x}$$

therefore the total gradient becomes:

$$\frac{\partial R}{\partial w_k} = \alpha \left\{ E[\phi(\tilde{y}_k, \tilde{\theta}_k)\sigma'(\tilde{y}_k)\mathbf{x}] - \alpha \sum_{j\neq k} E[\phi(\tilde{y}_j, \tilde{\theta}_j)\sigma'(\tilde{y}_j)\mathbf{x}] \right\} \tag{3.18}$$

This equation performs a constrained minimization in which the derivative with regard to one neuron can become orthogonal to the sum over the derivatives of all other synaptic weights. Therefore, it demonstrates the ability of the network to perform an exploratory projection pursuit in parallel, since the minimization of the risk involves minimization of the neurons $1, \ldots, N$.

Chapter 4

Analysis of the fixed points for the BCM rule

Unlike Oja's model, which has a simple treatment, for the BCM model it's worth doing an in-depth study of the fixed points. To do so we will now assume that the input vector can be chosen among many, contained in matrix $\mathbf{X}$. This one is a $m \times n$ matrix, where each of the m rows is a different kind of measure and each of the n columns is a different sample of the data, as described in section 2.1.
We also define the probability matrix $\mathbf{P}$, a diagonal $n \times n$ matrix, where the i-th diagonal element describes the probability of having the i-th input vector.

The neural activation function ϕ is, as before:

$$\phi(y, \theta) = y(y - \theta)$$

but now θ must be expressed according to the input probability:

$$\theta = E[y^2] = \sum_{i=0}^{n} p_i (\mathbf{x}_i \cdot \mathbf{w})^2 \tag{4.1}$$

where $\mathbf{x}_i$ is obviously a column of matrix $\mathbf{X}$.
The output equation is now:

$$\mathbf{y} = \mathbf{X}^T \mathbf{w} \tag{4.2}$$

where $\mathbf{y}$ is a $n \times 1$ vector. Then the equation 3.9 becomes:

$$\frac{d\mathbf{w}}{dt} = \dot{\mathbf{w}} = \mathbf{X}\mathbf{P}\boldsymbol{\phi} \tag{4.3}$$

where $\boldsymbol{\phi}$ is intended as a column vector of activation functions, one for every different n output: $\phi_i = y_i(y_i - \theta)$.

4.1 Linearly independent ambient

We now want to describe, as an example for the linearly independent ambient, the bidimensional case and then generalize it to higher dimensons.

4.1.1 Bidimensional case

In this case we have a single linear neuron operating in a bidimensional space (with $m = 2$ synapses) that receives $n = 2$ different inputs, the $\mathbf{X}$ and $\mathbf{P}$ matrix then are defined as:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix} \qquad \mathbf{P} = \begin{bmatrix} p_1 & 0 \\ 0 & p_2 \end{bmatrix}$$

while neuronal response and activation function are given by:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \qquad \boldsymbol{\phi} = \begin{bmatrix} y_1(y_1 - \theta) \\ y_2(y_2 - \theta) \end{bmatrix}$$

The components of vector $\dot{\mathbf{w}}$, calculated from the general equation (4.3), are then:

$$\begin{aligned} \dot{w}_1 &= y_1(y_1 - \theta)p_1 x_{11} + y_2(y_2 - \theta)p_2 x_{12} \\ \dot{w}_2 &= y_1(y_1 - \theta)p_1 x_{21} + y_2(y_2 - \theta)p_2 x_{22} \end{aligned} \tag{4.4}$$

The purpose is to find the fixed points, or those points that satisfy the equation:

$$\dot{\mathbf{w}} = 0 \tag{4.5}$$

It can be noted from equation (4.3) that this is only valid if $\boldsymbol{\phi} = 0$, as input vectors are assumed linearly independent and then the determinant of the correspondent matrix $\mathbf{X}$ is not zero ($\det \mathbf{X} \neq 0$). From this we can rewrite the condition (4.5) as:

$$\begin{aligned} y_1(y_1 - (p_1 y_1^2 + p_2 y_2^2)) &= 0 \\ y_2(y_2 - (p_1 y_1^2 + p_2 y_2^2)) &= 0 \end{aligned} \tag{4.6}$$

Solving this equation it is possible to find the fixed points of the system, given by the following possible output vectors:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \left\{ (0,0), \left(\frac{1}{p_1}, 0\right), \left(0, \frac{1}{p_2}\right), \left(\frac{1}{p_1 + p_2}, \frac{1}{p_1 + p_2}\right) \right\} \tag{4.7}$$

By stability analysis, it is possible to determine that only those in the form $\left(\frac{1}{p_1}, 0\right)$ or $\left(0, \frac{1}{p_1}\right)$ are stable (and are then points of maximum selectivity).
The stable solutions of vector $\mathbf{w}$ are then found by the inverse transformation of (4.3):

$$\mathbf{w} = (\mathbf{X}^T)^{-1}\mathbf{y} \tag{4.8}$$

4.1.2 K-dimensional case

We will now generalize the linearly independent case to higher dimensions, so our neuron now has $m = K$ synapses and can receive $n = K$ different inputs.
The neuronal dynamics, described in equation (4.3), appears now as:

$$\begin{bmatrix} \dot{w_1} \\ \dot{w_2} \\ \vdots \\ \dot{w_K} \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1K} \\ x_{21} & x_{22} & \dots & x_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ x_{K1} & x_{K2} & \dots & x_{KK} \end{bmatrix} \begin{bmatrix} p_1 & 0 & \dots & 0 \\ 0 & p_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & p_K \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_K \end{bmatrix} \tag{4.9}$$

As in the bidimensional case, the stationary condition (4.5) implies $\boldsymbol{\phi} = 0$, and the equations to find the fixed points are then:

$$y_i \left[y_i - \left(\sum_{j=0}^{K} p_j y_j^2 \right) \right] = 0 \quad i = 1, \dots, K \tag{4.10}$$

which leads to finding the following fixed points:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{bmatrix} = \left\{ \begin{array}{l} (0, 0, \dots, 0) \\ \left(0, 0, \dots, \dfrac{1}{p_i}, \dots, 0\right) \\ \left(0, 0, \dots, \dfrac{1}{p_i + p_j}, \dots, \dfrac{1}{p_i + p_j}, \dots, 0\right) \\ \left(0, 0, \dots, \dfrac{1}{p_i + p_j + p_k}, \dots, \dfrac{1}{p_i + p_j + p_k}, \dots, \dfrac{1}{p_i + p_j + p_k}, \dots, 0\right) \\ \vdots \\ (1, 1, \dots, 1) \end{array} \right\} \tag{4.11}$$

By stability analysis it's found, as before, that only the fixed points $\left(0, 0, \dots, \frac{1}{p_i}, \dots, 0\right)$ are stable, and the solutions of vector $\mathbf{w}$ are found by the same equation (4.8).

4.2 Linearly dependent ambient

We now want to extend the theory to the linearly dependent case which is the most probable in Data Analysis. Consider a neuron with $m = K$ synapses that can receive $n = N$ different inputs, where $N > K$ to ensure linear dependence.

Then the matrix $\mathbf{X}$ is now $K \times N$:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & \dots & x_{1N} \\ x_{21} & x_{22} & \dots & \dots & x_{2N} \\ \vdots & \vdots & \dots & \ddots & \vdots \\ x_{K1} & x_{K2} & \dots & \dots & x_{KN} \end{bmatrix} \tag{4.12}$$

while vector $\mathbf{w}$ is $K \times 1$ and vector $\boldsymbol{\phi}$ is $N \times 1$.

Deriving from respect to time the output equation:

$$\mathbf{y} = \mathbf{X}^T\mathbf{w}$$

and replacing in it the synaptic weight equation:

$$\dot{\mathbf{w}} = \mathbf{X}\mathbf{P}\boldsymbol{\phi}$$

we can now obtain a new neuronal activity rule:

$$\dot{\mathbf{y}} = \mathbf{X}^T\mathbf{X}\tilde{\boldsymbol{\phi}} \tag{4.13}$$

where $\tilde{\boldsymbol{\phi}} = \mathbf{P}\boldsymbol{\phi}$ is written for simplicity. The stability condition now can be imposed by requiring that neuronal output doesn't change over time, or $\dot{\mathbf{y}} = 0$.
This is satisfied if:

$$\mathbf{X}^T\mathbf{X}\tilde{\boldsymbol{\phi}} = 0 \tag{4.14}$$

but, being $\mathbf{X}$ a matrix of linearly dependent inputs, the equation has different solutions apart from the trivial one $\tilde{\boldsymbol{\phi}} = 0$: the kernel of $\mathbf{X}^T\mathbf{X}$ has then a dimension $Q \neq 0$.
As a consequence the rank of $\mathbf{X}^T\mathbf{X}$ has a dimension $W = N - Q$, where N is the dimension of matrix $\mathbf{X}^T\mathbf{X}$, from the linear algebra dimension theorem.
We can then define a basis for the kernel composed by Q vectors:

$$\mathcal{B}_{\ker(\mathbf{X}^T\mathbf{X})} = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_Q\}$$

and it is possible to write the first W components of $\boldsymbol{\phi}$ as a function of the remaining Q components with the system:

$$\begin{gathered} \phi_1 = v_{11}\phi_N^{(1)} + v_{21}\phi_{N-1}^{(2)} + \dots + v_{Q1}\phi_{W+1}^{(Q)} \\ \phi_2 = v_{12}\phi_N^{(1)} + v_{22}\phi_{N-1}^{(2)} + \dots + v_{Q2}\phi_{W+1}^{(Q)} \\ \vdots \\ \phi_W = v_{1W}\phi_N^{(1)} + v_{2W}\phi_{N-1}^{(2)} + \dots + v_{QW}\phi_{W+1}^{(Q)} \end{gathered}$$

that can be written down in a simpler form as:

$$\phi_i = \sum_{j=1}^{Q} v_{ji}\phi_{N+1-j}$$

From that, thanks to $\phi_i = y_i(y_i - \theta)$, the output equation is:

$$y_i^2 - y_i\theta - \sum_{j=1}^{Q} v_{ji}\phi_{N+1-j} = 0$$

with solution:

$$y_i = \frac{1}{2}\left(\theta \pm \sqrt{\theta^2 + 4\sum_{j=1}^{Q} v_{ji}\phi_{N+1-j}}\right) \qquad i = 1, \dots, W \tag{4.15}$$

These are then the form of the fixed points for the linearly dependent case.

4.2.1 Bidimensional neuron in a fourth-dimensional ambient

It is now presented, as an example, the case in which $K = 2$ synapses and $N = 4$ different inputs. The input matrix $\mathbf{X}$ is then:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \end{bmatrix}$$

while the output and neural activation are:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \qquad \boldsymbol{\phi} = \begin{bmatrix} y_1(y_1 - \theta) \\ y_2(y_2 - \theta) \\ y_3(y_3 - \theta) \\ y_4(y_4 - \theta) \end{bmatrix}$$

It is now possible to find a base for the kernel:

$$\mathcal{B}_{\ker(\mathbf{X}^T\mathbf{X})} = \{\mathbf{v}_1, \mathbf{v}_2\}$$

where we assumed that $Q = 2$ for simplicity (as it can also be 3 or 4). The $W = 2$ components of $\boldsymbol{\phi}$ can therefore be expressed as:

$$\begin{aligned} \phi_1 &= v_{11}\phi_3 + v_{21}\phi_4 \\ \phi_2 &= v_{12}\phi_3 + v_{22}\phi_4 \end{aligned}$$

Finally, the output stable solutions of the (4.15) are:

$$\begin{aligned} y_1 &= \frac{1}{2}\left(\theta \pm \sqrt{\theta^2 + 4v_{11}\phi_3 + 4v_{21}\phi_4}\right) \\ y_2 &= \frac{1}{2}\left(\theta \pm \sqrt{\theta^2 + 4v_{12}\phi_3 + 4v_{22}\phi_4}\right) \end{aligned} \tag{4.16}$$

From this it can be also be noted that it's possible to come back to the linearly independent case by assuming that the neuron does not responds to certain inputs:

$$\begin{cases} y_1 = \frac{1}{2}\left(\theta \pm \sqrt{\theta^2 + 4v_{11}\phi_3 + 4v_{21}\phi_4}\right) \\ y_2 = \frac{1}{2}\left(\theta \pm \sqrt{\theta^2 + 4v_{12}\phi_3 + 4v_{22}\phi_4}\right) \\ y_3 = y_3 \\ y_4 = y_4 \end{cases} \rightarrow \begin{cases} y_1 = \frac{1}{2}\left(\theta \pm \theta\right) \\ y_2 = \frac{1}{2}\left(\theta \pm \theta\right) \\ y_3 = 0 \\ y_4 = 0 \end{cases}$$

and the possible fixed points are exactly:

$$y = \left\{ (0,0,0,0), \left(\frac{1}{p_1}, 0, 0, 0\right), \left(0, \frac{1}{p_2}, 0, 0\right), \left(\frac{1}{p_1}, \frac{1}{p_2}, 0, 0\right) \right\}$$

as written in section 4.1.

4.2.2 Using the SVD to find a base for the kernel and as a pseudoinverse

A method is now propose to find both a base for the kernel (and then the components v_{ji} for the output equation) and an inverse formula for the weight vector.

Writing the input matrix through the Singular Value Decomposition $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$, it must be remembered in fact that the columns of $\mathbf{V}^T$ are the normalized eigenvectors of $\mathbf{X}^T\mathbf{X}$. Then the components v_{ji} of the kernel basis can be taken exactly from these eigenvectors, obviously only those with zero eigenvalue.
The SVD also does not only provide the components for the output equation, but also furnish a valid inverse equation:

$$\mathbf{w} = \mathbf{U}\Sigma^{-1}\mathbf{V}^T\mathbf{y} \tag{4.17}$$

generalizing the equation (4.8) to the linearly dependent case.

Chapter 5

Conclusions

After an in-depth analysis of the Oja's and BCM models, it's now possible to rapidly summarize the principal analogies and differences between them.

The model

From a general point of view, they both are artificial neural network models formed by perceptrons, that follow an unsupervised learning algorithm based on the Hebbian model. They however differ in the goal:

- Oja's rule extracts the principal components through diagonalizations of the covariance matrix

  $$\mathbf{C} = \langle \mathbf{x}^T \mathbf{x} \rangle$$

  thanks to eigenvector decomposition;

- BCM rule instead maximizes the selectivity of a neuron

  $$s(\mathbf{w}) = 1 - \frac{E[\mathbf{x}^T \mathbf{w}]}{\max[\mathbf{x}^T \mathbf{w}]}$$

  so it only responds to certain inputs.

The learning algorithm

Generally speaking, any learning algorithm can be reconducted to the simple rule:

$$\frac{d\mathbf{w}}{dt} = \phi(y)\mathbf{x} - \epsilon \mathbf{w}$$

but, while in BCM $\epsilon \to 0$ and $\phi(y) = y(y - \theta)$, in the Oja's rule $\epsilon = y^2$ and $\phi(y) = y$. Furthermore, both rules set a limit to the weights growth: BCM by introducing a sliding threshold θ and Oja by synaptic normalization.

Biological aspects

While the BCM model is entirely based on mathematical modeling of neurons of the visual cortex, Oja's model has nothing to do with biological aspects. We can in fact observe that, when extending Oja's rule to a network of artificial neurons, the algorithm is required to be non-local to correctly extract the principal components, proving that it is not based on reality.

Statistical aspects

Both models are very important in finding useful projections with statistical significance. The BCM model allows to extract relevant structures from high dimensional data distributions while Oja's model extracts the principal components. The two learning algorithms are then very similar from this point of view except for the BCM that, being extendeable to a non linear model, has lesser sensibility to outliers.

Bibliography

[1] A. Nielsen, *Neural Networks and Deep Learning*, Determination Press (2015)

[2] P. Dayan, L. F. Abbott, *Theoretical Neuroscience - Computational and Mathematical Modeling of Neural Systems*, MIT Press (2001)

[3] C. Giraud, *Introduction to High-Dimensional Statistics*, CRC Press (2014)

[4] H. Abdi, L. J. Williams, *Principal Component Analysis*, (2010)

[5] J. Shlens, *A Tutorial on Principal Component Analysis*, (2014)

[6] E. Oja, *Oja learning rule*, Scholarpedia (2008)

[7] E. Oja, *A Simplified Neuron Model as a Principal Component Analyzer*, (1982)

[8] T. Elze, *Modeling of a Recurrent Neural Network for Principal Component Analysis*, (2003)

[9] T. Sanger, *Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network*, (1989)

[10] E. L. Bienenstock, L. N. Cooper, P. W. Munro, *Theory for the Development of Neuron Selectivity: Orientation Specificity and Binocular Interaction in Visual Cortex*, (1981)

[11] N. Intrator, L. N. Cooper, *Objective Function Formulation of the BCM Theory of Visual Cortical Plasticity: Statistical Connections, Stability Conditions*, (1992)

[12] L. N. Cooper, N. Intrator, B. S. Blais, H. Shouval, *Theory of Cortical Plasticity*, World Scientific (2004)

[13] L. Acquaviva, *Soluzione del Modello di Apprendimento BCM in Presenza di Ambiente Generalizzato*, (2015)

[14] G. C. Castellani, N. Intrator, H. Shouval, L. N. Cooper, *Solutions of the BCM Learning Rule in a Network of Lateral Interacting Nonlinear Neurons*, (1999)