Maximization of Information Transfer in Neural Channels
University of Cape Town

Dean Rance

January 25, 2016

Abstract

Infomax is considered, and it is shown that it can be performed locally using hebbian and anti-hebbian learning, and hence is a biologically feasible optimization rule.

Moreover, not only do its consequences align with experimental data, but are also useful in artificial contexts.

Dedication

To Candace, as always.

Declaration

This is work I researched pretty much on my own. By finding other peoples' work.

Acknowledgments

I want to thank a lot of people for helping me through 2015.

For fear of forgetting someone, no names are mentioned. From academics to kith and kin, you all know who you are.

Notation

Throughout this text we use the following notation:

- X, Y denote random variables.
- $\mathbb{P}(X = x)$ denotes the probability of the outcome $X = x$.
- $\mathbb{E}(\cdot)$ is the expectation operator.
- $\langle \cdot \rangle$ denote trial or ensemble averages.
- $\mathcal{P}(z)$ is some probability distribution dependent on z.
- $p_X(x)$ or $p(x)$ denotes the pdf or pmf of X.
- $H(X)$ denotes the entropy of a variable X, while $H(p)$ denotes the entropy of a distribution p.
- $D_{KL}(p\|q)$ denotes the Kullback-Leibler divergence between distributions p and q.
- $I(X, Y)$ denotes the mutual information between variables X and Y.
- s is used to denote a stimulus, $\boldsymbol{s}$ a stimulus vector.
- $\boldsymbol{x}$ denotes the input to a channel.
- $\boldsymbol{y}$ denotes the input to a channel.
- h_i is used to denote the post-synaptic potential of a neuron indexed by i.
- f is used to denote a transfer function, f_i when needing to distinguish between neurons.
- W denotes the feedforward weight matrix, with $\boldsymbol{w}_i^T$ the vector which is W's i^{th} row.
- M denotes the lateral weight matrix.
- α and η denote learning rates.
- ν the Greek letter is used to denote output noise of a channel or neural network.
- Φ is used to denote a neuron or a noisy channel.

Contents

Chapter 1

Introduction

The layout of neurons in the brain is highly similar across different members of the same species; however, being incredibly complex leads one to believe that the genome cannot possibly determine this structure.

Consequently there must be something else which causes this structure. This phenomenon is thus caused self-organization, whereby relatively simple units, through some method, organize themselves into a complex whole. The neurons then, following some inherent properties, must develop the way they do.

These inherent properties can be formalized by an optimization principle which the neurons try to achieve, that is, an optimization principle guiding this development. One such principle is infomax, the assertion that collections of neurons should try to maximize the mutual information between their inputs and their outputs. This optimization principle is studied here.

It will be shown that the implications, should infomax be the correct principle, are consistent with observations of neural behaviour. Moreover, there is a learning mechanism which implements infomax, and finally that this learning mechanism is observed and biologically feasible.

We will conclude then that infomax is a viable candidate for explaining neuronal self-organization.

In order to understand infomax, we will first need to address the theoretical backgrounds of neural networks and information theory. Chapters 2 and 3 discuss the essentials of these, respectively. Chapter 4 elaborates on infomax in context, and gives some results. Chapter 5 considers principal component analysis (PCA), how it can be performed using infomax, its predictions, and how these predictions are consistent with observations in the brain. Chapter 6 does likewise, but for independent component analysis (ICA), an extension of PCA.

Chapter 2

Elements of Neural Networks

2.1 Structure of Neurons

In this section we consider the structure of neurons and neural networks insofar as is needed for later parts. Artificial neural networks will be described for the purpose of making analyses of neural networks tractable [For a more thorough exposition, see 8].

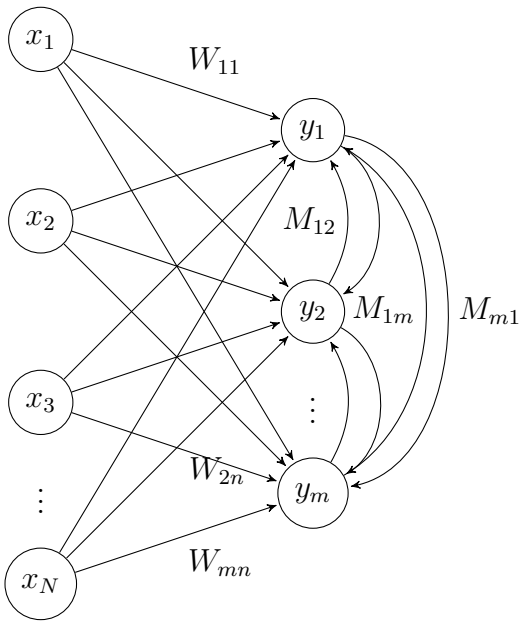


Figure 2.1: A typical 2 layer neural network with lateral connections.
Arrows denote direction of AP flow along axons. Not all synaptic efficacies W_{ij} are shown for readability.

Neurons are made up of several relevant parts:

- A cell body, enclosed in a membrane across which is a potential, called the membrane potential or post-synaptic potential (PSP). If this potential rises highly enough, the neuron fires an action potential (AP), or spike.

- Action potentials are charged pulses, which essentially embody the passing of information between neurons. After a neuron fires an AP, there is a brief

period, called the absolute refractory period, wherein the neuron cannot fire again.

- Dendrites which have receptor sites for neurotransmitters, which upon reception allow the opening and closing of ion channels. Ions flow through these channels, increasing or decreasing the membrane potential.
- Axons, along which APs flow. At the end of the axon are vesicules storing neurotransmitters, which are released into the synapse when the AP reaches the axon's end.
- Synapses, although not strictly part of neurons, are the small spaces between the axon endings and dendrite receptor sites.

Local learning putatively occurs at the synapses, where receptor sites, their sensitivities, and the amount of neurotransmitters released is altered. These properties are lumped together into what are called synaptic efficacies, or synaptic weights. In general, learning is performed by changing the properties of the neurons. They are essentially plastic, in that their number and sensitivity of receptor sites, length of axons, number of synapses, amount of neurotransmitter in the end of the axon, and other features, can all change - and do change, in response to the environment.

There are many features of neurons which are not typically considered in high-level models. These are simplifications made for computational purposes, or so that analytical solutions may be found. An example is that the inputs to a neuron are not necessarily additive - they may be subadditive or superadditive. That is, the downstream effects of different receptor sites receiving neurotransmitters is not necessarily the sum of the effects of each individual receptor site receiving neurotransmitters. However, additivity is a functional approximation. It is worth keeping in mind that most attempts and dealing with learning in neurons only select a few features of the neurons to address at any given time to allow tractability.

2.2 Firing Rates

Action potentials must convey their information through the timing of their firing: although the APs vary in shape, duration, and amplitude, they are all treated as identical instantaneous spikes.

Hence a sequence of n APs can be characterized by their firing times, t_i, $1 \leq i \leq n$, where the sequence is observed over some finite interval of time, $[0, T]$, so that $\forall i \; t_i \in [0, T]$.

We can define the neural response function as

$$\rho(t) = \sum_{i=1}^{n} \delta(t - t_i) \tag{2.1}$$

where δ is the Dirac delta function.

Because the exact sequence of APs in response to a given stimulus varies from trial to trial, neuronal responses are typically treated statistically. Sometimes they are characterized by "firing rate", which could have one of several interpretations:

- spike count rate is the number of APs that occur during a trial divided by trial length:
$$\boldsymbol{r} = \frac{n}{T} = \frac{1}{T}\int_0^T \rho(t)dt \tag{2.2}$$

- for a time dependent firing rate, we compute averages over short intervals of length Δt. However, because of the absolute refraction, for Δt small enough these averages will be 0 or 1. Hence the average is computed across several trials. Denoting the trial average by $\langle\cdot\rangle$, we get the trial averaged neural response function $\langle\rho(t)\rangle$, and the time dependent firing rate is
$$\boldsymbol{r}(t) = \frac{1}{\Delta t}\int_t^{t+\Delta t} \langle\rho(\tau)\rangle d\tau \tag{2.3}$$
Observe that this value is always between 0 and 1. Hence, $\boldsymbol{r}(t)\Delta t$ approximates the probability of an AP occurring in the interval $[t, t+\Delta)$

- For well-behaved integrals, we can replace the trial averaged neural response function with the time-dependent firing rate $\boldsymbol{r}(t)$:
$$\int h(\tau)\langle\rho(t-\tau)\rangle d\tau = \int h(\tau)\boldsymbol{r}(t-\tau)d\tau \tag{2.4}$$

- Spike count firing rate can be averaged across trials
$$\langle\boldsymbol{r}\rangle = \frac{\langle n\rangle}{T} = \frac{1}{T}\int_0^T \langle\rho(\tau)\rangle d\tau = \frac{1}{T}\int_0^T \boldsymbol{r}(t)dt \tag{2.5}$$
to get the average firing rate. These are the values we consider to be approximating, when using rate models.

If we put $\langle\boldsymbol{r}\rangle = f(s)$ where s is a fixed stimulus, then f is called the neural response tuning curve. It makes sense to compute this when the stimulus is held steady across several trials.

2.3 Types of Codes

Rate codes have the information encoded in the rate at which the neuron fires. While this allows for a robust method of information transfer, as the observed rate would centralize around a mean, the intended rate (given zero-mean noise), a consequence of this is that post-synaptic neurons cannot interpret the information until they have observed enough action potentials to ascertain the mean. Observe that

this value can be restricted to being between 0 and 1, by the discussion above. As a consequence, spike codes tend to allow for faster transfer of information.

Spike codes incorporate the actual timing of the spikes. Because the precise timing is subject to noise, this is often modeled by measuring whether the neuron fires within small time bins. As a consequence of the absolute refractory period, for small enough intervals, the neurons will fire at most once in any interval. Other spike models include encoding the information in the times between consecutive spikes, or synchrony across different neurons (which can sometimes interrupt one another's effects on a given post-synaptic neuron).

2.3.1 The Problem of Noise

Tuning curves allow us to predict average firing rates $f(s)$ for a stimulus s, but they do not describe deviation from the mean $\langle \boldsymbol{r} \rangle = f(s)$ (hence the pre-synaptic neuron has to fire for long enough for the post-synaptic neuron to estimate $\langle \boldsymbol{r} \rangle$). While the map f from stimulus to average firing rate may be deterministic, it is "likely that single trial responses such as spike count rates" $\boldsymbol{r}$ "can only be modeled probabilistically"[5].
Trial-to-trial deviation of $\boldsymbol{r}$ from $\langle \boldsymbol{r} \rangle$ is considered noise; models accommodating for this are thus called noise models.

If the standard deviation of the noise distribution is independent of $f(s)$, the variability is called additive noise; otherwise multiplicative noise (for example) has a standard deviation proportional to $f(s)$ (such as the Poisson distribution).

2.4 Properties of the Visual System

For comparing predictions with actual observations, we discuss the basic structure of visually responsive neurons [For a more thorough discussion, see 13, 5].

The process begins in the retina, converting light to AP sequences. The output neurons of the retina form the optic nerve conducting these AP sequences to the lateral geniculate nucleus (LGN) of the thalamus. From here, new signals are formed and passed onto the primary visual cortex (V1).

Neurons in these three regions respond to light stimuli in restricted regions of the visual field, called receptive fields (RFs). Within these receptive fields are regions where greater illumination enhances neuronal firing, and regions where diminished illumination enhances firing. These regions are arranged differently for different neurons, effectively allowing different neurons to be sensitive to different inputs. "Receptive fields", the term, can also refer to this spatial arrangement.

Retinal cells and LGN cells respond best to circular spots of light surrounded by darkness, or circular spots of darkness surrounded by light. These are their dominant receptive fields. In V1,"many neurons respond best to elongated light or dark bars, or boundaries between light and dark regions"[5]. We will see that these are the principal components, and indeed the independent components, of natural images.

2.5 Artificial Neural Networks

We will consider artificial neural networks in the rate coding paradigm i.e. the assumption that the outputs of the transfer functions are firing rates. The basic architecture of these networks are as follows:

- There is a layer of input neurons, denoted $\boldsymbol{x}$, simply called the inputs. The input is a vector $\boldsymbol{x} = (x_1, \ldots, x_N)$.

- There is a second layer of neurons, denoted $\boldsymbol{y} = (y_1, \ldots, y_M)$. Each such neuron has a weight vector $\boldsymbol{w}_i$ and a bias $\boldsymbol{w}_{0,i}$ used to compute the PSP $h_i = \boldsymbol{x}^T \boldsymbol{w}_i + \boldsymbol{w}_{0,i}$. These vectors are combined into a weight matrix W with weight vectors transposed as its rows.

- The PSP of each neuron in the second layer can also be affected laterally by other neurons, determined by a weight matrix M, so that for the i^{th} neuron $h_i \leftarrow \boldsymbol{x}^T \boldsymbol{w}_i + \boldsymbol{w}_{0,i} + \sum_{j \neq i} M_{ij} h_j$.

- Finally, each neuron in the second layer has a transfer function f_i so that its output is $y_i = f_i(h_i)$. Note that this can also be formalized to allow the transfer function to be computed before the lateral updates. There is no necessarily explicit order to these computations.

2.6 Locality and Hebbian Learning

Since neurons in the brain have only local information (no homunculus), for the purpose of biological realism our algorithms and learning methods require this property. Locality is where the learning update that a neuron performs uses only information known to the neuron at the times - it cannot, for example, know the PSP of another neuron. We consider two such related types of learning below.

2.6.1 Basic Hebbian Learning

Donald Hebb proposed an update rule of correlated behaviour of neurons leading to the mantra "neurons that fire together, wire together". Indeed the discovery of long-term potentiation in the hippocampus has provided evidence supporting the presence of this type of learning [13, For a discussion of Linsker's early results in deriving the receptive fields of LGN and V1 from simulations, see].

Hebbian learning is modeled as a learning rule whereby the learning updates depends on the product of the strengths of simultaneous activation of the pre- and post-synaptic neurons. Given an input $\boldsymbol{x}$ and a post-synaptic neuron y_i with weight vector $\boldsymbol{w}^i$ and transfer function f so that $y_i = f(\sum_j x_j w_{ij}) = f(\boldsymbol{x}^T \boldsymbol{w}_i)$, the learning rule can be written as

$$\Delta w_{ij}(t) = \alpha x_j y_i \tag{2.6}$$

$$\text{where} \quad w_{ij}(t) = \Delta w_{ij}(t) + w_{ij}(t-1) \tag{2.7}$$

where t denotes the timestep and $w_{ij}(t)$ is the value of the weight at timestep t, and α is a positive constant or independent variable called the learning rate. So, assuming α is the same for each weight and letting f be the identity function, for the full weight vector we get

$$\Delta \boldsymbol{w}_i^T(t) = \alpha(\boldsymbol{w}_i(t-1))^T(\boldsymbol{x}\boldsymbol{x}^T) \tag{2.8}$$

From this process we get

$$\Delta W(t) = \alpha(W(t-1))(\boldsymbol{x}\boldsymbol{x}^T) \to \alpha(W(t-1))C \tag{2.9}$$

where C is the autocorrelation matrix for the inputs $\boldsymbol{x}$ (defined as $C = \langle \boldsymbol{x}\boldsymbol{x}^T \rangle$). Strictly speaking for analogue systems (and so biological systems) we can perform the following transformation: setting $\alpha = \alpha' \times \Delta t$ where Δt is the timestep, we can write

$$\Delta W(t) = \alpha' \Delta t(W(t-\Delta t))(\boldsymbol{x}\boldsymbol{x}^T) \tag{2.10}$$

$$\text{as } \frac{\Delta W(t)}{\Delta t} = \alpha'(W(t-\Delta t))(\boldsymbol{x}\boldsymbol{x}^T) \tag{2.11}$$

$$\text{to get } \frac{d}{dt}W(t) = \alpha'(W(t))C \text{ as } \Delta t \to 0 \tag{2.12}$$

where differentiation is performed on each individual entry of the matrix. However, since all simulations run are performed digitally, we will not consider this form further.

2.6.2 Stable Hebbian Learning

It is straightforward to observe that this learning mechanism is unstable. Consider for example the right hand side of the above equation.

$$\alpha WC = \alpha \begin{pmatrix} \boldsymbol{w}_1^T C^1 & \boldsymbol{w}_1^T C^2 & \dots & \boldsymbol{w}_1^T C^n \\ \boldsymbol{w}_2^T C^1 & \boldsymbol{w}_2^T C^2 & \dots & \boldsymbol{w}_2^T C^n \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{w}_m^T C^1 & \boldsymbol{w}_m^T C^2 & \dots & \boldsymbol{w}_m^T C^n \end{pmatrix} \tag{2.13}$$

where C^i denotes the i^{th} column of the autocorrelation matrix C. Observe that, since C is positive semi-definite, every entry of WC is greater than or equal to zero. This means that at each learning step, w_{ij} grows for each i, j such that $(WC)_{ij} = \boldsymbol{w}_i^T C^j \neq 0$. This means that the basic hebbian learning mechanism has a positive feedback loop, and can grow without bound. So, unless there is some way of limiting the strengths of the synaptic efficacies, which is biologically plausible[1], the weights will increase without bound.

It is possible to renormalize the weights after each learning step so that the weight vectors corresponding to individual neurons have norm 1. Although this may be feasible and interesting, keeping all the weights on the surface of a sphere, and is

[1] If this were not to happen, it may induce an unpleasant phenomenon called excitotoxicity, essentially destroying the neuron as well as neighbouring and downstream neurons.

performed by some neural networks, it is nevertheless computationally expensive. So other methods have been sought.

A popular option is to use a decay term, which simulates loss due to imperfect memory. Ideally it would disallow the weights from having too great a magnitude, while keeping them fairly constant for a given post-synaptic neuron.

Miller and MacKay [14] performed quite an exhaustive study of these decay terms, and categorized them into two distinct types: multiplicative constraints and subtractive constraints.

Multiplicative constraints are of the form

$$\Delta W(t) = (W(t-1))C - \gamma(W(t-1))(W(t-1)) \tag{2.14}$$

$$\text{or} \quad \Delta(\boldsymbol{w}_i(t))^T = (\boldsymbol{w}_i(t-1))^T C - \gamma(\boldsymbol{w}_i(t-1))(\boldsymbol{w}_i(t-1))^T \tag{2.15}$$

where γ is a function of W or $\boldsymbol{w}_i$. The subtrahend can be viewed as limiting the growth of synaptic efficacies in proportion to the magnitude of the efficacies already in place.

Subtractive constraints are of the form

$$\Delta W(t) = (W(t-1))C - \gamma(W(t-1))\boldsymbol{v} \tag{2.16}$$

$$\text{or} \quad \Delta(\boldsymbol{w}_i(t))^T = (\boldsymbol{w}_i(t-1))^T C - \gamma(\boldsymbol{w}_i(t-1))\boldsymbol{v} \tag{2.17}$$

where $\boldsymbol{v}$ is a constant vector or a vector independent of W and $\boldsymbol{w}_i$..

2.6.3 Anti-Hebbian Learning

Where hebbian learning was originally proposed to use correlations, this uses anti-correlations. This type of learning is typically instantiated laterally i.e. within the same layer, and is modeled after interneurons in the brain which engage in inhibitory transmission of signals. The learning rule can be written as follows:

$$\Delta M_{ij} = -\eta\langle y_i y_j \rangle \tag{2.18}$$

where η is once again some positive constant or variable.

The update rules are stable, since there is negative, rather than positive, feedback in the algorithm. The updates become zero when the corresponding neurons have uncorrelated behaviour.

Chapter 3

Elements of Information Theory

In this chapter we review some fundamental concepts of information theory. Information theory was developed to answer the questions of what the optimal rate of transmitting information is across a (noisy) channel, what maximal compression of data is possible, and how to achieve these. Neurons can be regarded as a special case of noisy channels, and consequently information theory has been found useful for analyzing the transmission of messages throughout neural networks. It is in information theory that the infomax principle is founded.

3.1 Noisy Channels

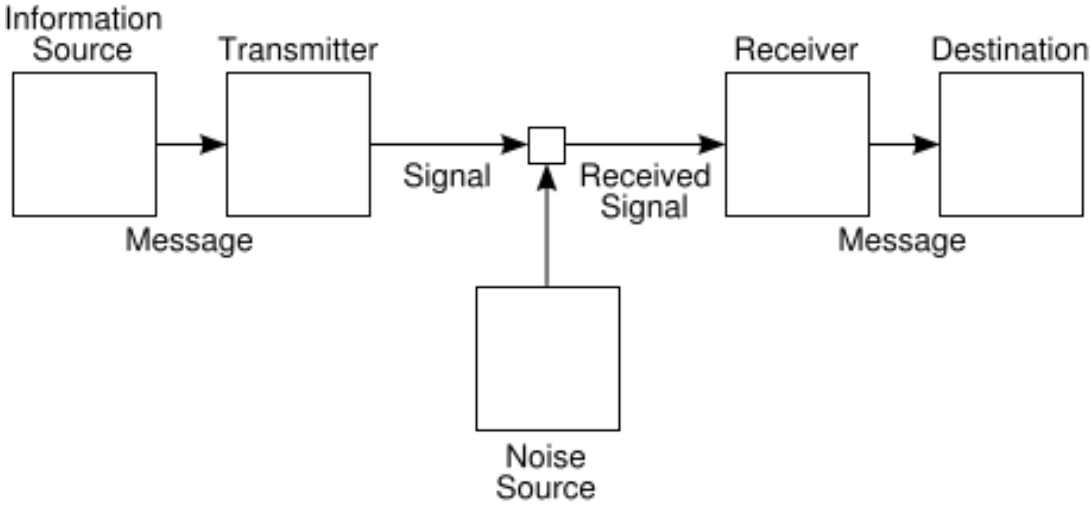


Figure 3.1: Claude Shannon's Noisy Channel Model
Taken From `http://market2science.eu/wp-content/uploads/2015/02/Shannon_communication_system.png`

A noisy channel is composed of three parts: a transmission variable or message $X = X(\omega)$ representing a message ω (X is called the codeword), a channel Φ which adds noise or corrupts X. and a received variable $Y = \Phi(X)$. The received variable is generated by a distribution dependent on the transmitted variable. The goal of information theory is to find the optimal representations of the transmitted variable so as to maximize the accuracy of inference of the message variable given the output

variable i.e. given Y we want to choose X, or the codewords, so as to minimize the error in inferring X from Y. Naturally, one has to consider the distribution of the variable X.

As an example, additive white Gaussian noise (AWGN) with strength T (measured in variance) gives $Y \sim \mathcal{N}(X, T)$ or $Y = X + \nu$ where the noise $\nu \sim \mathcal{N}(0, T)$. Here additive noise is simply noise added to the intended codeword X.

There can also be multiplicative noise, where the distribution of the noise (more than just the mean) is itself a function of the transmission variable i.e. $N \sim \mathcal{P}(f(X))$ where $\mathcal{P}$ is some probability distribution dependent on $f(X)$. An example of this may be where N is generated by a Poisson distribution with mean $f(X)$. Intuitively here one wants to choose $f(X)$ so as to minimize it for frequent values of X, and thus minimize the average effect of noise on the output.

Bayesian methods can then be used to infer $\mathbb{P}(X = x|Y = y)$ for some $x \in \{0, 1\}$, and $y \in \mathbb{R}$.

3.1.1 Neural Networks as Noisy Channels

A neuron can be regarded as a noisy channel. Given a stimulus s, a neuron Φ computs $\Phi(s)$. This is typically binary ($\Phi(s) \in \{0, 1\}$ i.e. the neuron either fires or does not) or a real variable between zero and one i.e. $\Phi(s) \in [0, 1]$. This is because of the absolute refractory period which, at small enough timescales, causes the neuron to be able to only fire once or not at all. Hence the firing rate at these scales is between 0 and 1. Artificial neurons have no such hard constraints.

Considering a sequence of neurons $\Phi_1, \Phi_2, \ldots, \Phi_n$ feeding consecutively into one another, and an input X, noise can cause the actual outputs of the neurons, $y_1, \ldots, y_n$, to not be the same as the computed values. For example, at the first layer we may get $y_1 \sim \mathcal{P}(\Phi_1(X))$, $y_2 \sim \mathcal{P}(\Phi_2(y_1))$ and so on, where $|phi$ here computes the desired noise-free output. Here $\mathcal{P}(\cdot)$ denotes probability distributions which might only depend on the argument for their mean i.e. additive noise.

That which is noise at the output of the one neuron, called output noise, is then input noise for the next neuron. A neuron in this architecture may potentially handle both noises. It can try infer the correct input, and adjust its inherent features to maximize the accuracy of this inference, and it can adjust its outputs to make such inference at later layers more accurate. This, or any combination of these, is called learning.

3.2 Shannon Information and Entropy

Information theory, as originally developed by Claude Shannon [19], has been applied with considerable success to the study of neural networks, both biological and artificial [6, 4]. Intuitively, the Shannon information (or just information) of the outcome of a random variable measures the amount of information one gets from such an outcome. Naturally then, a constant variable should give no information,

since its outcome can be predicted.

Given a discrete random variable X over $\Omega \subset \mathbb{R}$ with the probability mass function p_X, we can define the (Shannon) information of an output x as

$$h(x) := \log \frac{1}{\mathbb{P}(X = x)} \tag{3.1}$$

$$= \log \frac{1}{p_X(x)} \tag{3.2}$$

$$= -\log p_X(x) \tag{3.3}$$

where $\mathbb{P}(\cdot)$ denotes the probability of an event occurring, implicitly with respect to the appropriate probability measure denoted generically by $\mathbb{P}$.

The entropy of X is defined as:

$$H(X) := -\sum_{x \in \Omega} p_X(x) \log p_X(x) \tag{3.4}$$

$$= \left\langle \log \frac{1}{p_X(x)} \right\rangle_{p_X} \tag{3.5}$$

$$= \mathbb{E}\left(\log \frac{1}{p_X(x)} \right) \tag{3.6}$$

where $\langle \cdot \rangle_{p_X}$ is the average with respect to the distribution induced by p_X, and $\mathbb{E}(\cdot)$ is the expectation operator. Henceforth, except where needed for disambiguation, we shall drop the subscript X for a probability mass or density function p_X of a variable X. Occasionally the distribution p_X is used as the input for the entropy operator, and other entropy-related operators i.e. $H(p_X) = H(X)$

Observe that if $p(x_0) = 1$ for some $x_0 \in \Omega$ and zero everywhere else, then $H(X) = 0$ i.e. constant variables offer no uncertainty.

Here, we used the convention that $0 \times \log 0 = 0$. The base of the logarithm determines the units: when computed with logarithm base 2, the units are bits. However, it is often more convenient to use the natural logarithm, in which case the units are nats. Base 10 logarithms have units of digits.

Notice that nowhere is the actual value of x used for the information of the outcome $X = x$. A trivial consequence of this is translational invariance.

The entropy of a variable is also the average length of a codeword representing the variable (in binary, say, if $\log_2$ is used). It goes without saying that shortening the length of the average codeword shortens the time taken to transmit codewords.

3.3 Joint and Conditional Entropies

Let X, Y be random variables over Ω_X, Ω_Y respectively. Let the joint probability mass function be $p(x, y)$ and the conditional probability of $Y = y$ given $X = x$ be

$p(y|x)$. Then the joint entropy of X and Y is

$$H(X,Y) := -\sum_{x\in\Omega_X}\sum_{y\in\Omega_Y} p(x,y)\log p(x,y) \tag{3.7}$$

$$= \mathbb{E}\left(\frac{1}{p(x,y)}\right) \tag{3.8}$$

$$= \left\langle \log\frac{1}{p(x,y)} \right\rangle_{p(x,y)} \tag{3.9}$$

and the conditional entropy of Y given X is

$$H(Y|X) := \sum_{x\in\Omega_X} p(x)H(Y|X=x) \tag{3.10}$$

$$= -\sum_{x\in\Omega_X} p(x)\sum_{y\in\Omega_Y} p(y|x)\log p(y|x) \tag{3.11}$$

$$= -\sum_{x\in\Omega_X}\sum_{y\in\Omega_Y} p(x,y)\log(y|x) \tag{3.12}$$

(by the chain rule for probabilities) which is the uncertainty of Y averaged over all concrete outcomes of X. These definitions can be extended to provide the entropy of a random vector $\boldsymbol{X} = (X_1,\ldots,X_n)$ as the joint entropy of its components $H(\boldsymbol{X}) = H(X_1,\ldots,X_n)$, which are all random variables in their own right.

3.4 Kullback-Leibler Divergence

An important concept is that of Kullback-Leibler divergence, also known as relative entropy or cross-entropy. It is a measure of the "difference" between two distributions p and q given by

$$D_{KL}(p\|q) := \sum_{x\in\Omega} p(x)\log\frac{p(x)}{q(x)} \tag{3.13}$$

This is not a metric, as $D_{KL}(p\|q) \neq D_{KL}(q\|p)$ in general. However, we do have

$$D_{KL}(p\|q) \geq 0 \qquad \text{and} \qquad D_{KL}(p\|q) = 0 \iff p = q \tag{3.14}$$

This can be seen using Jensen's inequality:

$$\begin{aligned} -D_{KL}(p\|q) &= \sum_{x\in\Omega} p(x)\log\frac{q(x)}{p(x)} \\ &\leq \log\left(\sum_{x\in\Omega} p(x)\frac{q(x)}{p(x)}\right) \\ &= \log 1 = 0 \end{aligned}$$

since log is a concave function.

3.5 Mutual Information

Mutual information is used to measure statistical independence between variables X, Y with associated distributions $p(x), p(y)$. The mutual information can be characterized as the Kullback-Liebler divergence between the joint distribution $p(x, y)$ and the factorized ones:

$$I(X,Y) := D_{KL}(p(x,y)\|p(x)p(y)) \tag{3.15}$$

$$= \sum_{x\in\Omega_X}\sum_{y\in\Omega_Y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} \tag{3.16}$$

This is clearly symmetric: $I(X,Y) = I(Y,X)$. We also have that $I(X,X) = H(X)$ and if X and Y are independent, then $I(X,Y) = 0$. Thus mutual information is a measure of the amount of information one variable conveys about another.

Hence, if X is the input, Y the output of a noisy channel, then $I(X,Y)$ is a measure of the amount of information conveyed by the channel.

Some useful properties of mutual information are its non-negativity, which follows from the non-negativity of Kullback-Leibler Divergence, and reformulations

$$I(X,Y) = H(X) - H(X|Y) \tag{3.17}$$

$$= H(Y) - H(Y|X) \tag{3.18}$$

$$= H(X) + H(Y) - H(X,Y) \tag{3.19}$$

which follow from the fact that the operand of the logarithm in the definition $\frac{p(x,y)}{p(x)p(y)}$ is the same as $\frac{p(x|y)}{p(x)}$ by the chain rule for probabilities.

From this we observe that $I(X,Y) \leq \min(H(Y), H(X))$.

3.6 Redundancy

If $\boldsymbol{X} = (X_1, \ldots, X_n)$ is a random vector, then the mutual information of its components provides a measure of redundancy $R(\boldsymbol{X})$ of the vector:

$$R(\boldsymbol{X}) := I(X_1, \ldots, X_n) \tag{3.20}$$

$$= D_{KL}\left(p(x_1, \ldots, x_n)\|p(x_1)\ldots p(x_n)\right) \tag{3.21}$$

$$= \sum_{x_1\in\Omega_{X_1}} \cdots \sum_{x_n\in\Omega_{X_n}} p(x_1, \ldots, x_n)\log\frac{p(x_1, \ldots, x_n)}{p(x_1)\ldots p(x_n)} \tag{3.22}$$

This is minimized when the vector $\boldsymbol{X}$ is factorized.

3.7 Extension to Continuous Distribution

The definitions for discrete random variables can be generalized to continuous random variables X with probability density functions p_X. Since confusion is unlikely,

the same notation will be used throughout for continuous and discrete variables, as well as with the corresponding information theoretic concepts. Below, however, for clarity, we will let $\tilde{H}(\cdot)$ denote the entropy of a discrete random variable.

Consider a discrete random variable X assuming values $x_k = k\delta x$ where $k \in \mathbb{Z}$ with associated probabilities $\hat{p}(x_k)$. As long as $\infty > \delta x > 0$, we have

$$\tilde{H}(X) = -\sum_{k=-\infty}^{\infty} \hat{p}(x_k) \log \hat{p}(x_k), \qquad \sum_{k=-\infty}^{\infty} \hat{p}(x_k) = 1 \tag{3.23}$$

For the continuous limit, we define the probability density $p_X(x_k) = \hat{p}(x_k)/\delta x$. Then

$$\tilde{H}(X) = -\sum_{k=-\infty}^{\infty} \delta x p_X(x_k) \log p_X(x_k) + \log(1/\delta x), \qquad \sum_{k=-\infty}^{\infty} \delta x p_X(x_k) = 1 \tag{3.24}$$

Notice that, provided the following integrals exist, we get

$$\lim_{\delta x \to 0} \sum_{k=-\infty}^{\infty} \delta x p_X(x_k) \log p_X(x_k) = \int p_X(x_k) \log p_X(x_k) dx \tag{3.25}$$

and

$$\lim_{\delta x \to 0} \sum_{k=-\infty}^{\infty} \delta x p_X(x_k) = \int p_X(x_k) dx = 1 \tag{3.26}$$

However, $\lim_{\delta x \to 0} \log(1/\delta x) = \infty$. Because of this, we define the differential entropy to be

$$H(X) := -\int p_X(x) \log p_X(x) dx \tag{3.27}$$

There is nothing particularly foreign about this concept. If we let n denote the counting measure, and m the usual Lebesgue measure, then the probability mass function of a discrete random variable outputting values in some set $A \subset \mathbb{R}$ can be written as

$$\sum_{x \in A} p_X(x) = \int_A p_X dn \tag{3.28}$$

while for a continuous random variable Y this would simply be

$$\int_A p_Y dm \tag{3.29}$$

Similarly, we get

$$H(X) = \int_\Omega p_X \log p_X dn \quad \text{and} \quad H(Y) = \int_\Omega p_Y \log p_Y dm \tag{3.30}$$

3.7.1 Uniform Distribution

First we compute the entropy of the uniform distribution: let $q = \frac{1}{b-a}$ be the uniform distribution on an interval $[a, b]$. Then

$$H(p) = -\int_a^b p(x) \log p(x) dx \tag{3.31}$$

$$= \int_a^b \frac{1}{b-a} \log(b-a) dx \tag{3.32}$$

$$= \frac{1}{b-a} \times [x \log(b-a)]_a^b \tag{3.33}$$

$$= \log(b-a) \tag{3.34}$$

We get from the positivity of Kullback-Leibler divergence that, if $p(x)$ and $q(x)$ are two pdfs on an interval $[a, b]$, then

$$0 \leq \int_a^b p(x) \log \frac{p(x)}{q(x)} dx \implies -\int_a^b p(x) \log q(x) dx \geq -\int_a^b p(x) \log p(x) dx \tag{3.35}$$

Now let p be any probability mass function on $\{x_1, \ldots, x_n\}$, and let $q(x_i) = \frac{1}{n}\ \forall i$. Then

$$-\sum_{i=1}^n p(x) \log q(x) = \sum_{i=1}^n \log n = \log n \tag{3.36}$$

which we see is the entropy of q. Hence $H(p) \leq H(q)$. The generalization to continuous distributions follows from the previous method of rewriting the probability mass function $p(x)$ to the probability density function $p(x)\delta x$. Hence, for a fixed interval, we see that the uniform distribution has the maximum entropy.

3.7.2 Gaussian Distribution

Here we will compute the differential entropy of a Gaussian random variable X, firstly for a single-valued Gaussian with mean μ and variance σ^2. The pdf is

$$p(x) = \frac{\exp(-\frac{1}{2}[x-\mu]^2/\sigma^2)}{\sigma\sqrt{2\pi}} \tag{3.37}$$

The entropy is therefore

$$\begin{aligned}
H(X) &= -\int_{\mathbb{R}} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \log\left(\frac{\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)}{\sqrt{2\pi\sigma^2}}\right) dx \\
&= -\int_{\mathbb{R}} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \left(-\frac{(x-\mu)^2}{2\sigma^2} - \log\sqrt{2\pi\sigma^2}\right) dx \\
&= \log\sqrt{2\pi\sigma^2} \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) + \frac{1}{2\sigma^2} \int_{\mathbb{R}} \frac{(x-\mu)^2}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \\
&= \log\sqrt{2\pi\sigma^2} + \frac{\sigma^2}{2\sigma^2} \\
&= \frac{1}{2}\left(\log(2\pi\sigma^2) + 1\right)
\end{aligned}$$

Similarly, for a variable $\boldsymbol{X}$ with multivariate Gaussian distribution of dimension n with mean $\boldsymbol{\mu}$ and covariance matrix Σ, and m the Lebesgue measure, we get

$$\begin{aligned}
H(\boldsymbol{X}) &= -\int_{\mathbb{R}^n} p(\boldsymbol{x}|\boldsymbol{\mu},\Sigma)\log\left(\frac{\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)}{(2\pi|\Sigma|)^{n/2}}\right)dm(\boldsymbol{x}) \\
&= -\int_{\mathbb{R}^n} p(\boldsymbol{x}|\boldsymbol{\mu},\Sigma)\left(-\frac{n}{2}\log\left(2\pi|\Sigma|\right)\right)dm(x) \\
&\quad -\int_{\mathbb{R}^n} p(\boldsymbol{x}|\boldsymbol{\mu},\Sigma)\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)dm(x) \\
&= \frac{n}{2}\log\left(2\pi|\Sigma|\right) + \frac{1}{2}\int_{\mathbb{R}^n} p(\boldsymbol{x}|\boldsymbol{\mu},\Sigma)\mathrm{Tr}\left(\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^T\right)dm(x) \\
&= \frac{n}{2}\log\left(2\pi|\Sigma|\right) + \mathrm{Tr}\left(\Sigma^{-1}\frac{1}{2}\int_{\mathbb{R}^n} p(\boldsymbol{x}|\boldsymbol{\mu},\Sigma)(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^T dm(x)\right) \\
&= \frac{n}{2}\log\left(2\pi|\Sigma|\right) + \frac{1}{2}\mathrm{Tr}\left(\Sigma^{-1}\Sigma\right) \\
&= \frac{n}{2}\left(\log\left(2\pi|\Sigma|\right)+1\right)
\end{aligned}$$

Now, for a fixed variance σ^2, let $q(x)$ be a pdf with the Gaussian distribution, and let $p(x)$ be some arbitrary distribution with the same variance. Translational invariance of entropy allows us to assume that they both have the same mean μ. Now consider the Kullback-Leibler divergence of the two distributions:

$$\begin{aligned}
0 \le D_{KL}(p\|q) &= -H(p) - \int_{\mathbb{R}} p(x)\log q(x)dx \\
&= -H(p) - \int_{\mathbb{R}} p(x)\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)\right)dx \\
&= -H(p) - \int_{\mathbb{R}} p(x)\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)dx - \log e\int_{\mathbb{R}} p(x)\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)dx \\
&= -H(p) + \frac{1}{2}\log(2\pi\sigma^2) + \frac{\sigma^2}{2\sigma^2} \\
&= -H(p) + \frac{1}{2}\left(\log(2\pi\sigma^2)+1\right) \\
&= -H(p) + H(q)
\end{aligned}$$

where the definition of variance was used to get from the third line to the fourth. Rearrangement gives $H(p) \le H(q)$ and so, for a fixed variance, the Gaussian distribution has the greatest entropy.

3.8 Histogram Equalization

Since maximizing information transfer in a channel can be equated with maximizing the entropy of the output distribution, one finds that for a bounded transfer function setting the derivative of the transfer function of the channel equal to input's probability density function achieves optimal transfer of information, that is, the transfer function is the cumulative density function of the input distribution. This achieves a uniform output distribution over the range of the transfer function, which

maximizes the entropy of the output variable. So every output occurs with equal probability.

The intuition [9] is that if the sensitivities of the transfer function are too high for regular inputs, then the outputs are frequently saturated (in our case, the firing rate frequently is near 1. This is particularly bad since a Poisson model would predict greater noise here than elsewhere). If the sensitivities are too low, then large parts of the response range are underutilized because they correspond to unlikely inputs. So the inputs should be coded so that the response levels are used with equal frequency.

An alternative intuition is, given a small amount of additive output noise, we want to minimize the effect thereof. This is done by increasing the relative effect of noise on infrequent inputs so that one can decrease the effect of noise on frequent inputs.

An example of what may be histogram equalization may be the attention we pay to words: imagine, for arguments sake, that we paid more attention to words which occurred less frequently. If while listening to someone, they used some exotic word, we may pay more attention or be more aware of it than the usual syntactical words of English such as “of” or “the”. Then, if the amount of attention paid could be measured numerically, and we multiplied the frequency of a word occurring with the attention paid to it, histogram equalization would have occurred if this product were fairly uniform.

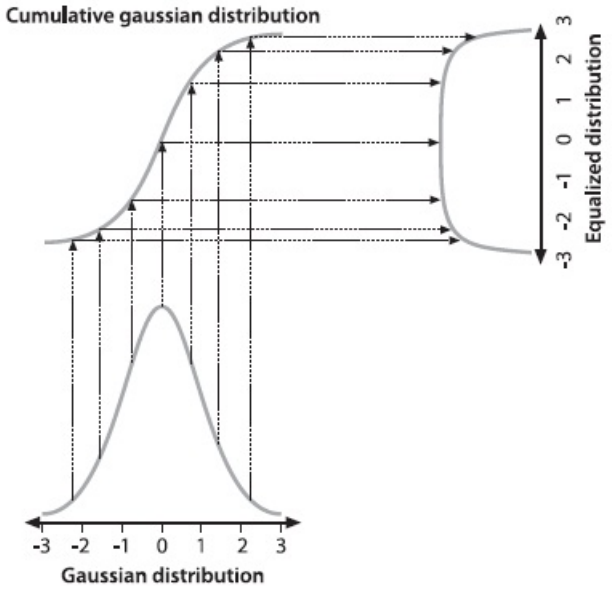
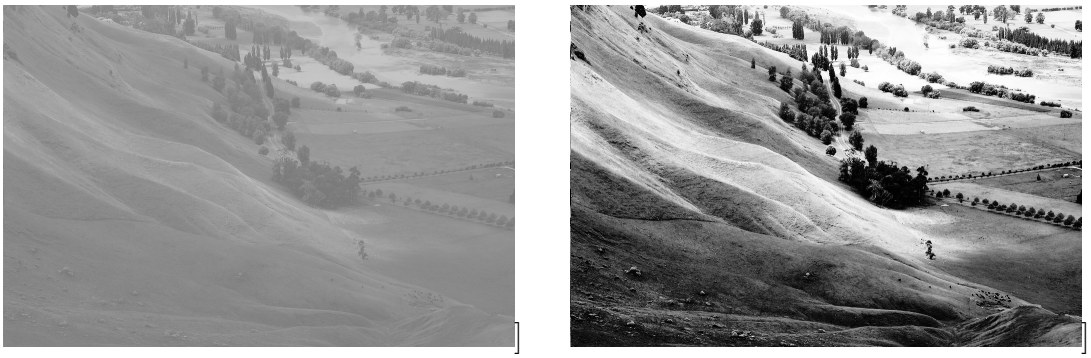


Figure 3.2: Gaussian Histogram Equalization Over Finite Interval
Taken from `http://sapachan.blogspot.co.za/2010/04/learning-opencv-histogram-equalization.html`



(a) Before Histogram Equalization.

(b) After Histogram Equalization.

Figure 3.3: An Example Of Histogram Equalization
Taken from `https://upload.wikimedia.org/wikipedia/commons/0/08/Unequalized_Hawkes_Bay_NZ.jpg` and `https://upload.wikimedia.org/wikipedia/commons/0/08/Eequalized_Hawkes_Bay_NZ.jpg` respectively.

Chapter 4

Infomax

In this section we outline discuss what infomax is, and the purpose it serves. In the next sections how it generalizes to perform PCA and ICA.

Nadal and Parga [17] point out that a systematic approach had been developed for analyzing sensory coding:

1. Decide on a task that may possible be fulfilled by the particular sensory system

2. Define an objective function that characterizes the performance of this system

3. Compute the optimal performance that could be obtained

4. Compare this performance with experimental data

It is assumed that sensory inputs are represented, or "encoded", by the neurons for efficient further processing. Hence some criteria are suggested are based on information theory.

Two such related criteria are redundancy reduction [1] and infomax [12]. Redundancy reduction is considered for comparison, as some interesting observations can be made by this comparison.

The need for some optimization function arises from the idea of self-organization. Given the complexity of the brain, yet the relative uniformity of its patterns across instances, it is assumed that it is neither encoded by the genome nor a random product of development. The neural structure must then organize with respect to some organization principle.

4.1 Redundancy Reduction

Barlow's proposal focuses on the readability of the presented representation, and hence to optimize for compression of data neurons should code for features statistically independent from other features. The optimal code that achieves this redundancy reduction is a factorial code. Deviation from a factorial code would then be penalized.

4.2 Infomax

Infomax is the principle that neurons should maximize the mutual information between the input and the output (sensory data and neural representation, respectively). While primarily a mechanism to describe handling of neuronal noise (indeed it is not well-defined in the absence of noise, as the differential entropy diverges!), it nevertheless turns out to, in the near absence of noise, perform other tasks well.

4.3 Results on Infomax

For feedforward networks, with non-linear transfer functions, Nadal and Parga [16] showed, in the low noise limit (so that the problem is well-defined), that infomax, when performed over both the synaptic efficacies and the choice of transfer functions, leads to a factorial code i.e. redundancy reduction.

As it turns out, mutual information will be maximum if

1. synaptic efficacies are such that PSPs are statistically independent

2. transfer functions for each cell are chosen according to histogram equalization

In fact, the transfer functions, when non-linear, allow the PSPs to pick up on higher order moments and perform more than simply decorrelation. This result is related to Blind Source Separation (BSS) (See chapter 6). That is, infomax can be used as a cost function for performing BSS [2].

Although only the rate-coding paradigm is within scope, it is worth noting that for feedforward networks with stochastic outputs, Nadal, Brunel and Parga [15] further showed the factorization result "remains valid in the limit of vanishing noise" whenever it is the distribution of the output, and not the output itself, which depends on a deterministic function of the input. Hence the result generalizes to spiking neurons.

So it turns out then that infomax also, where possible, enhances both the readability of the code and improves the transfer rate by minimizing redundancy. Ultimately, it is a hopeful candidate for explaining self-organization. The next result is an important one, as it characterizes what the non-linearities do in these non-linear networks.

4.4 Histogram Equalisation in Neurons

Laughlin [9] computed and verified the optimal function for transferring information for a single neuron, a large monopolar cell having a single input and a single output, in the blowfly's visual system. He computed that, to optimize information transfer, histogram equalization should be performed. This well-matched the experimental data which, as far as the approach mentioned before is concerned, is the optimal outcome. This experiment, although performed before infomax was proposed, verifies infomax in this special case.

Chapter 5

Extension to Principle Component Analysis

5.1 Principle Component Analysis

In this section we discuss the problem of principal component analysis (PCA). We will see that a hebbian learning implementation can be used to perform PCA and derive biologically relevant results. While there are many such implementations, we need only consider one for the thesis of this text. Moreover, infomax in some cases reduces to preforming PCA, including the discussed biologically relevant case. Ultimately, an understanding of PCA will also aid in the analysis of ICA as a consequence of infomax.

As it turns out, under the assumption of Gaussian distributed inputs, a neural network performing PCA maximizes the information transfer. In fact, derivation of the visual receptive fields, both in the LGN and V1, can be realized by preforming PCA on Gaussian white noise inputs. Linsker originally performed this derivation using Linsker networks (2-dimensional layers of neurons receiving inputs from neurons selected by a Gaussian distribution from neurons in the layer above, with mean directly above the neuron) and hebbian learning, and he equated it to the problem of PCA.

In principal component analysis, one is given observations from a mixture of distributions (or a multivariate distribution) and the goal is to find the principal components, i.e. the directions in which the data has most variance, and represent the data with respect to these components. This typically allows for effective compression of data, which in the case of a neural network, may allow for increased information transfer rate. As it turns out, these principal components are in fact the eigenvectors of the covariance matrix, or the correlation matrix if the results are to be found with respect to maximal correlation as opposed to maximal covariance.

We show that under Gaussian input distributions, infomax equates to performing PCA. Moreover, a linear neural network can perform PCA with hebbian and anti-hebbian learning and only local information. This can be done, for example, by having the hebbian learning finds the principal subspace, and asymmetric anti-hebbian learning finds a representation of this subspace with respect to the principal

components.

5.2 Why PCA?

Where possible, infomax "reduces" to performing ICA (see next chapter). This however is not always possible. One of the prominent instances is in the context of a multivariate Gaussian input.

Given that infomax is still defined in these circumstances, we need to consider what is actually achieved. A linear neural network disallows picking up higher order moments, and so performs PCA, achieving infomax on Gaussian input distributions.

Interestingly, infomax was originally equated with PCA, probably due to the fact that infomax appeared in literature before ICA! This may have set research back a few years, but allowed for the derivation of some interesting results.

Note that in general, a linear neural network cannot achieve maximum information transfer.

5.3 Hebbian PCA

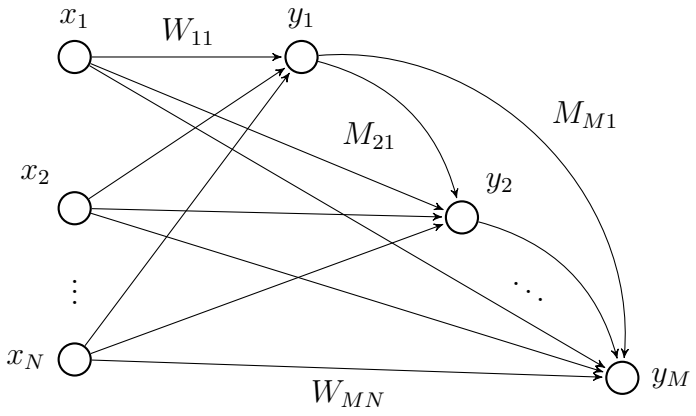


Figure 5.1: The neural network of the hebbian/anti-hebbian algorithm. Interneuron anti-hebbian connections denoted by matrix M, feedforward hebbian connections by W. $M \leq N$

In this section we present an algorithm put forth by Sanger [18] for performing PCA using a neural network with hebbian and anti-hebbian learning mechanisms.

The overall structure of the algorithm is relatively simple. Every output neuron y_i receives the entire range of inputs $\boldsymbol{x}$ and has multiplicative weight decay. However, each output neuron is calculated in turn, with each consecutive one receiving anti-hebbian responses from all previous output neurons. The result is that the first neuron learns the direction in which the data correlates the most, but is scaled quadratically so as to not have its overall weight deviate much from 1. The second neuron then finds the direction of maximal correlation of inputs while simultaneously

decorrelating itself from the first neuron, achieving ultimately complete decorrelation from the first neuron but maximal input correlation direction, hence the direction of the second PC. The third neuron does the same, but decorrelates itself from both previous neurons, and so forth. Provided the number of output neurons M is less or equal to the number of input neurons N, this algorithm finds the M principal components.

Sanger [18] developed an algorithm, the Generalized Hebbian Algorithm (GHA), which performs PCA. At the end, each output is the response to a single eigenvector of the correlation matrix, and the outputs are ordered by decreasing eigenvalues.

Let $\boldsymbol{x}$ be the inputs to a single layer linear network (two layers, including the input layer) with an $M \times N$ weight matrix W, and let the outputs be $\boldsymbol{y} = W\boldsymbol{x}$, with $M \leq N$.

If the values of $\boldsymbol{x}$ are generated by a stationary white random vector stochastic process, with correlation matrix $C = \langle \boldsymbol{x}\boldsymbol{x}^T \rangle$, then $\boldsymbol{x}$ and $\boldsymbol{y}$ are both time-varying, and so will be W as a result of the adaptation algorithm.

GHA is given by

$$W_{ij}(t+1) = W_{ij}(t) + \gamma(t) \left(y_i(t)x_j(t) - y_(t) \sum_{k<i} W_{ki}(t)y_k(t) \right) \tag{5.1}$$

or, in matrix form,

$$\Delta W(t) = \gamma(t) \left(\boldsymbol{y}(t)\boldsymbol{x}^T(t) - \mathrm{LT}\left(\boldsymbol{y}(t)\boldsymbol{y}^T(t)\right) W(t) \right) \tag{5.2}$$

where $\mathrm{LT}(\cdot)$ sets all elements above the diagonal to zero, making the argument a lower triangular matrix.

If $\gamma(t)$ is such that $\lim_{t\to\infty} \gamma(t) = 0$ and $\sum_{t=0}^{\infty} \gamma(t) = \infty$, then, Sanger proves, for random weights assigned at time 0, then the algorithm will converge, and W will converge to the matrix consisting of the first M eigenvectors of the correlation matrix, ordered by decreasing eigenvalue, with probability 1.

5.4 Linear Infomax and PCA

For a Gaussian input distribution, the reason is simple: decorrelation is the same as independence. Consequently, finding the independent components of the distribution equates to finding decorrelated directions of the distribution. Since nonlinear scaling does not occur (a bounded non-linear transfer function will force all the outputs to be between its bounds, potentially equalizing their histograms over this range, effectively transforming the Gaussian input ellipsoid level sets into a uniformly distributed hypersphere) the directions that are least correlated are exactly the eigenvectors of the correlation matrix, that is, the principal components.

5.5 Biological Relevance

The difference between the PCs found by applying PCA to natural images, and the receptive fields of the visual cortex, occur primarily in the minor components [7]. This is likely because the PCs were found under no noise conditions, and so the minor components differ from the redundant information encoded by the brain. That is, if the brain finds the principal components which it can, limited by input noise, and embodies them in visual fields, then the results of this mechanism is consistent with what is observed.

Chapter 6

Extension to Independent Component Analysis

6.1 Independent Component Analysis

Below we will demonstrate that independent component analysis (ICA), a generalization of PCA, can also be performed with a neural network by including nonlinear transfer functions. As it turns out, this can be done with only local information and hebbian/anti-hebbian learning. Consequently, this is a biologically feasible mechanism. The result is a higher order generalization of Laughlin's observation with the blowfly's neuron.

We first show that this is what would be expected as an outcome if infomax were implemented, and then that this outcome actually occurs.

Independent component analysis (ICA) is essentially a higher-order generalization of PCA. Where PCA considers only the variance of the data, ICA considers higher order moments. The assumption is that we have a linear mixture of potentially independent sources. We wish to find a representation of this information using maximally independent components.

The goal of independent component analysis is to find a representation of the data of independent components i.e. not simply decorrelate the components of a random vector but actually render them independent. As an intriguing observation, the implicit assumption of Gaussianity for PCA (i.e. that the independent components and the maximally variant orthogonal components are the same, or that maximizing variance maximizes independence) reveals a problem for non-Gaussian mixtures: the central limit theorem indicates that mixtures of distributions tend to be more Gaussian than not, and so PCA, which looks to maximize independence under the assumption of Gaussianity, actually tends to maximize the mixture rather than invert it.

The classic application for ICA is the "cocktail party" problem, or blind source separation. This entails reversing a linear mixture of independent sources. We have N sources, and N distinct observations such that the mixture is invertible. A related problem is blind deconvolution, whereby the effect of an unknown filter needs to be

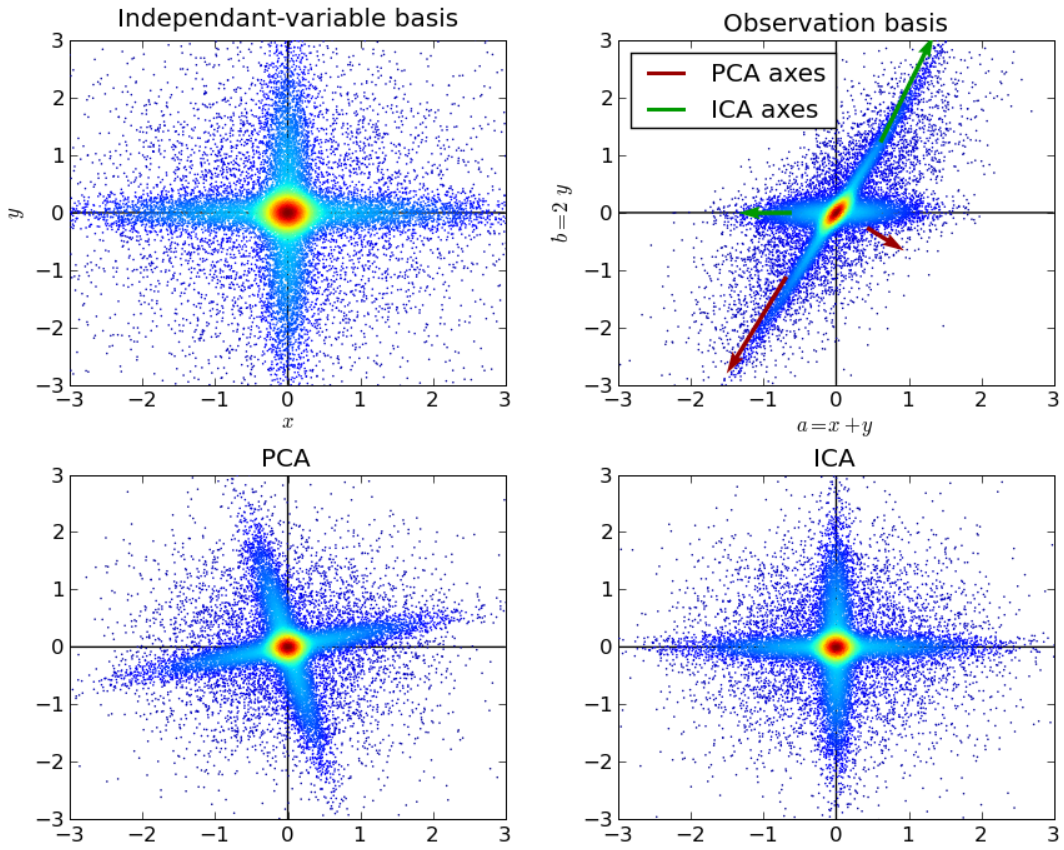


Figure 6.1: PCA, ICA comparison on Linear Mixture of Independent Variables. Taken from `http://gael-varoquaux.info/science/attachments/ica_pca/ica_on_non_gaussian_data.png`

reversed. These problems are typically used to test an ICA methodology, whereupon the method discussed below was successful.

6.2 Performing ICA

Bell and Sejnowski [2] observed that the generalization of histogram equalization (having the sloping part of the transfer function align with the peaks of the input distribution) to multiple inputs and outputs leads to a system which reduces redundancy between output units while maximizing information transfer. This is, in effect, ICA.

We present here their neural network algorithm, with non-linear transfer functions, for achieving exactly this. First let X be the input, Y the output of a 1 input 1 output neural network. We wish to maximize $I(Y,X) = H(Y) - H(Y|X)$. We assume there is no known noise (which we may have wished to filter out), and the neural network mapping $\Phi(X) = Y$ is deterministic, and $H(Y|X)$ has its lowest possible value, diverging to $-\infty$. To avoid the complexities of this divergence, where others have fixed an infinitismal noise for well-definedness, Bell and Sejnowski consider only the gradients since then the reference terms in the differential entropies causing this

divergence disappear.

If w is a parameter in the mapping Φ, then differentiating with respect to it gives

$$\frac{\partial}{\partial w} I(Y, X) = \frac{\partial}{\partial w} H(Y) \tag{6.1}$$

since $H(Y|X)$ does not depend on w. To see this, consider a system which avoids divergence by having small output noise:

$$Y = \Phi(X) + \nu \tag{6.2}$$

where Φ is an invertible transformation, ν the additive output noise. Then

$$H(Y|X) = H(\nu) \tag{6.3}$$

So maximizing the mutual information is equivalent to maximizing the output entropy $H(Y)$, because $\frac{\partial}{\partial w} H(\nu) = 0$.

Thus, for given input distributions and invertible differentiable deterministic Φ, mutual information is maximized by maximizing output entropy $H(Y)$.

We seek here effectively an algorithm which achieves Laughlin's observation of histogram equalization.

If Φ is strictly monotonically increasing, or decreasing, so as to have a unique inverse, then the output pdf $p_Y(y)$ can be written with respect to the input pdf $p_X(x)$ using the change of variables formula:

$$p_Y(x) = p_X(y^{-1}) \times \left| \frac{\partial x}{\partial y} \right| = \frac{p_X(x)}{|\partial y / \partial x|} \tag{6.4}$$

Thus, the output entropy is given by

$$\begin{aligned} H(Y) &= -\mathbb{E}(\log p_Y(y)) && (6.5)\\ &= -\int_{-\infty}^{\infty} p_Y(y) \log p_Y(y) dy && (6.6)\\ &= \int_{-\infty}^{\infty} p_Y(y) \log \left| \frac{\partial y}{\partial x} \right| dy - \int_{-\infty}^{\infty} p_X(x) \left| \frac{\partial x}{\partial y} \right| \log p_X(x) dy && (6.7) \end{aligned}$$

where the subtrahend on the right of the final line, the entropy of X, can for simplicity be considered unaffected by alterations of w in determining $\Phi(X) = Y$. So to maximize $H(Y)$, we need only consider the minuend, or first term, when changing w, which is the average log of how the input affects the output.

This can be done by using a sample of x's to approximate $p_X(x)$, and then using an online stochastic gradient ascent rule:

$$\begin{aligned} \Delta w &\propto \frac{\partial H}{\partial w} \\ &= \frac{\partial}{\partial w} \left(\log \left| \frac{\partial y}{\partial x} \right| \right) \\ &= \left(\frac{\partial y}{\partial x} \right)^{-1} \frac{\partial}{\partial w} \left(\frac{\partial y}{\partial x} \right) \end{aligned}$$

or simply

$$\Delta w = \alpha \left(\frac{\partial y}{\partial x}\right)^{-1} \frac{\partial}{\partial w}\left(\frac{\partial y}{\partial x}\right) \tag{6.8}$$

for some learning rate α.

For illustrative purposes we can use their example of the logistic transfer function, $y = \frac{1}{1+e^{-h}}, \quad h = wx + w_0$ where w_0 is a bias term, and w a weight. Computing their derivatives, we get

$$\frac{\partial y}{\partial x} = wy(1-y) \tag{6.9}$$

$$\frac{\partial}{\partial w}\left(\frac{\partial y}{\partial x}\right) = y(1-y)(1+wx(1-2y)) \tag{6.10}$$

Letting the first equation be the divisor, the second the dividend, we get

$$\Delta w \propto \frac{1}{w} + x(1-2y) = \frac{1}{w} + x\left(1 - \frac{2}{1+e^{-wx-w_0}}\right) \tag{6.11}$$

The same process with w_0 gives

$$\Delta w_0 \propto 1 - 2y \tag{6.12}$$

Observe that the Δw rule is anti-hebbian (subtracting the product of x and a function of y so that the input is being decorrelated with the output) with an anti-decay term $\frac{1}{w}$. The anti-hebbian term keeps y away from saturating values at 0 and 1 (as would be the case if the weights succumbed to a positive feedback loop and diverged, letting $h \to \pm\infty$), but the anti-hebbian rule alone would make the weight tend to 0 which would send h to $\frac{1}{2}$, so the anti-decay term $\frac{1}{w}$ keeps y away from 0 by growing significantly when w moves too close to 0. This balance of terms produces a distribution near uniform over the interval $(0, 1)$, in effect performing histogram equalization.

Now we consider an $N \to N$ network with input $\boldsymbol{x}$, weight matrix W, bias $\boldsymbol{w_0}$, and output $\boldsymbol{y}$. Then, similarly to before,

$$p_Y(\boldsymbol{y}) = \frac{p_X(\boldsymbol{x})}{|J|} \tag{6.13}$$

where $|J|$ is the absolute value of the Jacobian of the transformation from $\boldsymbol{x}$ to $\boldsymbol{y}$. Now, instead of maximizing $\log\left|\frac{\partial y}{\partial x}\right|$, we maximize $\log|J|$. Notice that this is the log of the volume of the space into which the values of $\boldsymbol{x}$ are mapped. We are effectively spreading the points uniformly.

Again using the logistic function $\boldsymbol{y} = \frac{1}{1+e^{-\boldsymbol{h}}}, \quad \boldsymbol{h} = W\boldsymbol{x} + \boldsymbol{w_0}$, similar rules can be derived:

$$\Delta W \propto \left(W^T\right)^{-1} + (\mathbf{1} - 2\boldsymbol{y})\boldsymbol{x}^T \tag{6.14}$$

$$\Delta w_0 \propto \mathbf{1} - 2\boldsymbol{y} \tag{6.15}$$

where **1** is the vector of ones. The anti-hebbian term is now an outer product, and the anti-decay term is now an anti-redundancy term (since the weight matrix tends to being singular if its rows or columns align, that is, if its outputs align).

For an individual weight w_{ij} we get

$$\Delta w_{ij} \propto \frac{\text{cof} w_{ij}}{\det W} + x_j(1 - 2y_i) \tag{6.16}$$

where $\text{cof} w_{ij}$ is the cofactor of w_{ij}. Because of the anti-redundancy term avoiding convergence on a singular matrix, when the weight vectors for distinct outputs become too similar, these updates cause then to diverge.

Linsker [10] observed that the computation above appears highly non-local, and hence biologically infeasible. Computing $\left(W^T\right)^{-1}$ appears to require a non-local computation.

He offered the following local method of computing this value, which, in conjunction with Bell and Sejnowski's algorithm above, allows an entirely local implementation for performing infomax.

Since we are computing the inverse transpose of feedforward weights, the transfer functions are irrelevant. So consider the equation $\boldsymbol{h} = W\boldsymbol{x}$. Let lateral connections between each pair of output units i, j be described by a matrix M. These are used recursively to compute an auxiliary vector

$$\boldsymbol{v}(t) = W\boldsymbol{x} + M\boldsymbol{v}(t-1), \quad \text{where} \quad \boldsymbol{v}(0) = W\boldsymbol{x}$$

for $t = 1, 2, \ldots$ timesteps. For a fixed $\boldsymbol{x}$, $\boldsymbol{v}(t)$ converges to $\boldsymbol{v} = W\boldsymbol{x} + M\boldsymbol{v}$ or $\boldsymbol{v} = (I - M)^{-1}W\boldsymbol{x}$ provided all eigenvalues of M have absolute value < 1 (letting the inverse be defined).

This network is used to compute $\left(W^T\right)^{-1}$. Define $q = \langle \boldsymbol{x}\boldsymbol{x}^T \rangle$ and $Q = \langle \boldsymbol{u}\boldsymbol{u}^T \rangle = WqW^T$, $\langle \cdot \rangle$ denoting the ensemble average. Assuming the ensemble vector $\boldsymbol{x}$ spans the input space, q is a positive definite matrix.

Assume the lateral weights M can be made to satisfy $M = I - \gamma Q$ (for a given W), γ chosen to ensure the eigenvalue condition. Linsker notes that we must then have $0 < \gamma < 2/\lambda$ where λ is the largest eigenvalue of Q. Then $Q^{-1} = \gamma(I - M)^{-1}$ and so $I = \gamma(I - M)WqW^T$, giving

$$\left(W^T\right)^{-1} = \gamma(I - M)^{-1}Wq = \gamma\langle \boldsymbol{v}\boldsymbol{x}^T \rangle \tag{6.17}$$

The component $W_{ij}^{-1} = \langle v_j x_i \rangle$ is computed using only local information.

A practical way of computing M is to let it evolve incrementally, according to

$$\Delta M = \eta(-\gamma \boldsymbol{h}\boldsymbol{h}^T + I - M) \tag{6.18}$$

where η is a learning rate. This is an anti-hebbian learning rule (from the $-\eta\gamma\boldsymbol{h}\boldsymbol{h}^T$ term) and it, too, is local, since

$$\Delta M_{ij} = \eta(-\gamma h_i h_j + \delta_{ij} - M_{ij}) \tag{6.19}$$

where δ_{ij} is the Kronecker delta.

A catch in the implementation is that an input $\boldsymbol{x}$ needs to be held steady at the inputs while the iterative computation takes place. Linsker "improved" on this [11] by finding another solution, but in a rate-coding paradigm it is entirely feasible that inputs remain present since they need to be in order for the post-synaptic neuron to ascertain the rate.

The linear output $u_j = \sum_i W_{ji} x_i$ is stored at each output unit j. Each pair of outputs h_j, h_k is used to compute ΔM_{jk} for the lateral connection from k to j. The outputs $v_j(t)$ are iteratively computed until asymptotic values v_j are obtained. Then finally for each feedforward connection i to j, x_i and v_j are used to compute $\left(W^T\right)^{-1}_{ji}$.

6.3 Predictions of Non-Linear Infomax

We now consider the predictions of a neural network, using infomax as an optimization criterion and not constrained by linearity. Effectively, this results in the network reducing redundancy in its outputs, and thus finding a maximally independent representation of its inputs. This, in effect, is independent component analysis.

6.4 Biological Relevance

Having shown that non-linear infomax can perform independent component analysis using only biologically available properties of locality and hebbian/anti-hebbian learning, it remains to show that the results thereof align with actual experimental observations.

That is exactly what has been found in [3]. Here, Bell and Sejnowski performed their algorithm on natural images and found that the ICA derived components present properties of the receptive fields of simple cells in the primary visual cortex, those of being localized and oriented bar and edge filters. One may wonder why both ICA and PCA find similar results. Indeed, it is entirely conceivable that both mechanisms take place. Xi, Jiangtao, et al. found [20] from simulations that Bell and Sejnowski's algorithm, under some circumstances, would not work well unless the data was preprocessed such as by PCA. This suggests that the mechanisms at play are more complex and layered than hoped, but nevertheless working in conjunction to achieve the result of maximizing information transfer.

Chapter 7

Conclusion

We have considered infomax as a possible optimization function to describe self-organization and learning. Its predictions have been verified by experiments, and it has been demonstrated that the process can be performed with local knowledge and hebbian and anti-hebbian learning.

Hence, although many different optimization functions lead to similar outcomes under various circumstances, we see that infomax is at the very least consistent.

Further research would then be to find different outcomes for different optimization functions, so as to differentiate these different optimization functions based on their predictions.

Furthermore, it must be noted that the models are not perfect: biological methods for performing the computations have not been considered here, nor has the property that excitatory synapses do not, as of yet observed, ever become inhibitory, or vice versa.

Bibliography

[1] Horace B Barlow. "The coding of sensory messages". In: *Current problems in animal behaviour* (1961), pp. 331–360.

[2] Anthony J Bell and Terrence J Sejnowski. "An information-maximization approach to blind separation and blind deconvolution". In: *Neural computation* 7.6 (1995), pp. 1129–1159.

[3] Anthony J Bell and Terrence J Sejnowski. "The "independent components" of natural scenes are edge filters". In: *Vision research* 37.23 (1997), pp. 3327–3338.

[4] Anthony CC Coolen, Reimer Kühn, and Peter Sollich. *Theory of neural information processing systems*. Oxford University Press, 2005.

[5] Peter Dayan and Laurence F Abbott. *Theoretical neuroscience*. Vol. 806. Cambridge, MA: MIT Press, 2001.

[6] Gustavo Deco and Dragan Obradovic. *An information-theoretic approach to neural computing*. Springer Science & Business Media, 2012.

[7] Peter JB Hancock, Roland J Baddeley, and Leslie S Smith. "The principal components of natural images". In: *Network: computation in neural systems* 3.1 (1992), pp. 61–70.

[8] Simon S Haykin et al. *Neural networks and learning machines*. Vol. 3. Pearson Education Upper Saddle River, 2009.

[9] Simon Laughlin. "A simple coding procedure enhances a neuron's information capacity". In: *Zeitschrift für Naturforschung c* 36.9-10 (1981), pp. 910–912.

[10] Ralph Linsker. "A local learning rule that enables information maximization for arbitrary input distributions". In: *Neural Computation* 9.8 (1997), pp. 1661–1665.

[11] Ralph Linsker. "Improved local learning rule for information maximization and related applications". In: *Neural networks* 18.3 (2005), pp. 261–265.

[12] Ralph Linsker. "Self-organization in a perceptual network". In: *Computer* 21.3 (1988), pp. 105–117.

[13] Josh McDermott. "The Emergence of Orientation Selectivity in Self-Organizing Neural Networks". In: *The Harvard Brain* 3.1 (1996), pp. 43–51.

[14] Kenneth D Miller and David JC MacKay. "The role of constraints in Hebbian learning". In: *Neural Computation* 6.1 (1994), pp. 100–126.

[15] Jean-Pierre Nadal, Nicolas Brunel, and Nestor Parga. "Nonlinear feedforward networks with stochastic outputs: infomax implies redundancy reduction". In: *Network: Computation in neural systems* 9.2 (1998), pp. 207–217.

[16] Jean-Pierre Nadal and Nestor Parga. “Nonlinear neurons in the low-noise limit: a factorial code maximizes information transfer”. In: *Network: Computation in neural systems* 5.4 (1994), pp. 565–581.

[17] JEAN-PIERRE Nadal and NESTOR Parga. “Sensory coding: information maximization and redundancy reduction”. In: *Neural information processing* 7.1A-2 (1999), pp. 164–171.

[18] Terence D Sanger. “Optimal unsupervised learning in a single-layer linear feedforward neural network”. In: *Neural networks* 2.6 (1989), pp. 459–473.

[19] Claude Elwood Shannon. “A mathematical theory of communication”. In: *ACM SIGMOBILE Mobile Computing and Communications Review* 5.1 (2001), pp. 3–55.

[20] Jiangtao Xi et al. “On the INFOMAX algorithm for blind signal separation”. In: *Signal Processing Proceedings, 2000. WCCC-ICSP 2000. 5th International Conference on*. Vol. 1. IEEE. 2000, pp. 425–428.