FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# Missing signal restoration by means of an entropy criterion

Recuperação de sinais perdidos aplicando um critério de entropia

Igor Vladimiro Agostinho Proença da Silva

Licenciado em Engenharia Electrotécnica e de Computadoress
pela Faculdade de Engenharia da Universidade do Porto

Dissertação submetida para satisfação parcial dos
requisitos do grau de mestre
em
Engenharia Electrotécnica e de Computadores

Dissertação realizada sob a supervisão de
Professor Doutor Vladimiro Henrique Barrosa Pinto de Miranda
do Departamento de Engenharia Electrotécnica e de Computaodres
da Faculdade de Engenharia da Universidade do Porto

Porto, Setembro de 2009

## Resumo:

Os principais objectivos da presente tese resumem-se ao treino de redes neuronais e avaliação do comportamento de um algoritmo de treino baseado no conceito de entropia correlacionada do erro aplicado ao problema de restauração de sinais perdidos ou corrompidos.

Para algoritmo de controlo, foi escolhido o bem conhecido e eficiente critério de minimização do erro médio quadrático sujeito às mesmas condições de treino.

A característica de auto-associação de informação das redes neuronais comummente denominadas auto-codificadores (auto-encoders) através do estabelecimento de suposta correlação entre partículas de informação aparentemente caóticas e não correlacionadas servirá de estrutura base para a recuperação de sinais. De facto, a existência desta correlação é absolutamente necessária para a recuperação de sinais perdidos. Tal implica a posse de informação suficiente sobre outros sinais conhecidos para tornar a reconstrução de sinais perdidos possível. Para avaliar as capacidades de compressão de informação dos auto-codificadores, usar-se-ão apenas auto-codificadores de camada intermédia única.

Uma aplicação prática da teoria é apresentada. Para tal, usou-se uma série temporal de leituras de vários sensores de uma central eólica anónima realizadas ao longo de um dado período. Para proceder à recuperação de sinais perdidos simulou-se a falha de vários sensores usando valores arbitrários para amostras consecutivamente adquiridas durante um período de tempo bem definido. Auto-codificadores separadamente treinados com ambos os critérios estabelecem a função auto-associativa que permitirá a recuperação dos sinais perdidos aplicando os métodos já mencionados.

Como se poderá comprovar ao longo do presente documento, o algoritmo baseado no critério de maximização da entropia correlacionada do erro ultrapassa em todos os aspectos o seu concorrente, demonstrando através das funções de densidade de probabilidade do erro das respectivas soluções a sua inequívoca superioridade.

## Abstract:

The present thesis main objectives are to train neural networks using entropy based concepts and evaluate missing signal restoration performance when error correlation entropy criterion is the chosen criterion.

Performance comparison will use for control algorithm the solutions found using the well known and effective mean square error minimization criterion when subject to the same signal recovery problems.

A particular type of neural network, commonly denominated auto-encoder, known auto-associative behaviour and its proven ability to find correlation between apparent decorrelated information particles will serve as the base structure needed to perform signal restoration. The existence of such correlation within information particles is essential to effectively recover corrupt signals, hence implying that some sufficient known information is needed to allow corrupted signal reconstruction. Only single hidden layer auto-encoders will be used to allow the evaluation of this layer's input compression capabilities.

One real world application is presented. A time series comprising various eolic power central sensor readings will be used to test missing signal recovery. To simulate missing sensors, randomly selected sensor readings were substituted by arbitrary values over a consecutive period of time. Separately error maximization correntropy and mean squared error minimization trained auto-encoders implement the auto-associative mapping that will allow missing signal restoration, enforcing such recovery either using one or the other already refered methods.

As one will show through the remainder of this document, both auto-encoding and signal restoration by means of an error correlated entropy maximization criterion has proven itself to be superior in all aspects, as one will conclude by thoroughly inspecting both algorithms solution's error probability density functions and other additional data.

# PREFACE

A portuguese popular saying claims that “Stormy weather always ends with sunny weather”.

Many personal and professional difficulties I have faced during this last year, but sunshine always shined at the end, with a little help from all of my friends.

So, acknowledgement to this special group of people could not be overlooked.

Luckily, all of you know who you are and I know you know your importance to me, even if I don't put your name in here.

Briefly, I would like to express my sincere gratitude to my parents, my sisters, all of my close ancestors for their imprint in the depths of my soul, to all of my friends and to my teacher for their contribution to make a better me.

# INDEX

# TABLE INDEX

# FIGURE INDEX

# Chapter 1

## Introduction

# 1 Introduction

To study and evaluate the performance of a theoretic information entropy based criterion and compare this method with the well known mean squared error method when applied to auto-encoder training and missing signal restoration are the main objectives of the herein reported studies.

Many approaches can be taken to solve an enormous variety of problems related to signal encoding and missing signal restoration (1).

Nowadays, signal encoding is found almost in any modern application, either result of the need to turn information unreadable for security reasons, either to ensure data readability and reliability, and most of the times, to ensure both conditions.

Such encoding will depend of the specific application and moreover of the signal characteristics and the need of it to comply with previously stated conditions, obviously.

Signal corruption is also a well known problem that affects numerous engineering and other applications, with several possible causes. Hopefully, for some applications it's possible to detect and eliminate the sources of such errors and prevent them to occur again in the future, normally using preventive actions, like preventive maintenance where the equipment is replaced as soon as it reaches the limit imposed by their lifetime expectancy. Some other, due to intrinsic disability of the equipment to run flawlessly for a given period of time, will soon or later generate erroneous and, sometimes, complete signal loss. And let us not forget the cases when different equipments used to measure the same signals under the same conditions result in incoherent and therefore meaningless information, if one can not tell which is the correct signal, if any.

Independently of the reasons causing signal corruption and the respective solutions adopted to prevent future loss of information, in some critical applications, restoration of previously acquired signals is imperative for the correct functioning of the system, as is the case of measuring the electric power delivered by any power central to customers. As one can see, in face of any discrepancies between the effectively produced power and the readings made by some power measurement equipment may result in clear prejudice of both customer and electric power producer.

These are the cases when signal encoding may reveal useful to recover corrupted signals, as long there is some kind of previous information about the phenomenon or if it's possible to establish some correlation between some other read signals that would allow its restoration.

Unfortunately, there are still some applications where the apparent chaotic or insufficient knowledge about the signal behaviour unfolds the impossibility of signal restoration using the nowadays available tools.

Within the conditions depicted previously, namely, sufficient signal behaviour knowledge and correlation, neural networks have already proven to be a good tool to elaborate studies about these questions.

Although, as stated in the previous paragraph, neural networks themselves aren't the main concern of this analysis.

In fact, throughout this document, the topology of the neural networks used is maintained according to the exemplifying neural network in figure 1.

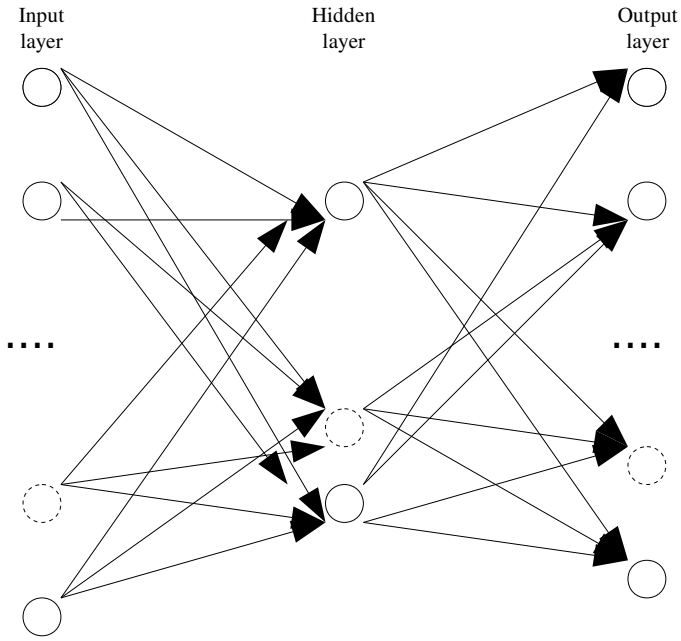


**Figure 1: Model for a single hidden layer auto-encoder**

This specific type of neural networks is commonly denominated auto-encoder.

In remembrance, this type of feed forward neural network encodes the signals that are put in its inputs and reflects them in its outputs.

Surely, this can only be done after proper network training by the use of some supervised learning algorithm. More details about neural network supervised training algorithms will follow in the subsequent paragraphs.

Then again, and specifically for the problem in hands, only single hidden layer auto-encoders were used, in accordance to the example shown in figure 1.

This was done with the clear purpose of evaluating the neural networks ability to compress information in the hidden layer and synapses that connect these neurons to the output ones.

The final objective of using strictly single hidden layer auto-encoders was to evaluate the capability of the auto-encoder to extract the maximum information of the data input and therefore narrowing down the data needed to perform in conformance to what is expected of an auto-encoder.

To do so, the size of the hidden layers used throughout the various studies was set to be as small as possible in order to not compromise intended goals. By size of a layer let us understand number of neurons that compose it.

To better explain this assumption, consider the diagram in figure 2.

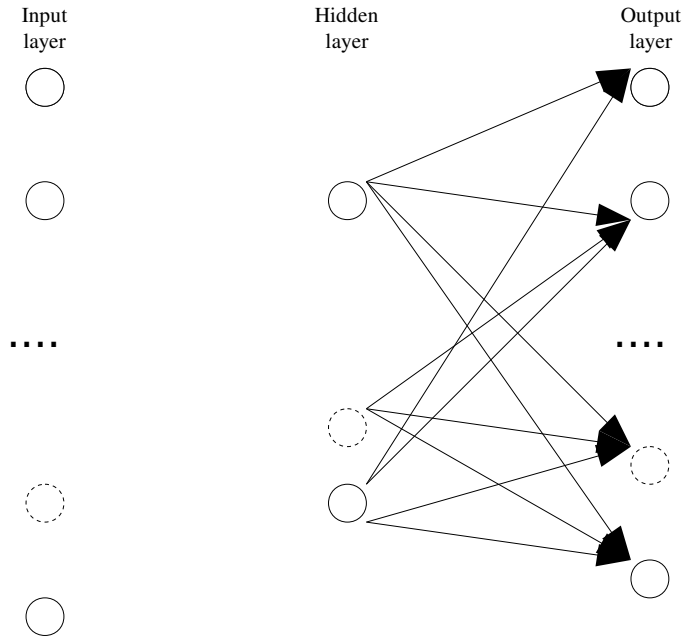


**Figure 2: A detached input or hidden-input auto-encoder**

As one can see in figure 2, the input has been detached from the hidden and output layers. Assuming that the auto-encoder has already been trained to perform as expected, it is possible to inject in the hidden layer neurons the respective input values to obtain the same outputs obtained using the complete network.

In this case, there is no need to store information regarding the input values and synapses that connect input and hidden layer neurons. Moreover, all the information required to perform in accordance to what is expected of an auto-encoder is now stored in the smaller sized hidden layer thus implying original information compression.

In terms of auto-encoder use and properties evaluation, this resumes what is expected as final outcome of the present study.

As foretold, such auto-encoding will need some supervised learning algorithm to perform as intended. On the other hand, to perform such learning, some criterion must be followed.

Since the training criteria adopted can be decoupled from the learning machine structure, these criteria are general learning schemes which can be used to map some desired output function (2).

In literature, the most widely used criterion to perform such training is the mean square error minimization. Specifically for the feed forward neural networks, training is done by back propagation. But good training results are only obtained when the estimated probability density function has a Gaussian behaviour (2). More on this subject can be found in chapter 3.

Problems related to mean square error minimization criterion lead to the development of the information theoretic learning criteria (2).

This theory goes around the intractability of Shannon's information entropy definition (3) offering an elegant approach to estimate information entropy for a given data set.

The definition of information potential and information forces conjugated with Parzen windowing non-parametric methods originate an estimate of the probability density function for a finite number of data sets easily computable, as one will see in the following chapter.

Once one can estimate the finite data set information entropy, as explained in chapter 2, one can now set up a new criterion based on the information entropy maximization or minimization to train some learning machine.

In fact, Xu in (2) states that maximizing the data set information entropy implicitly minimizes the error set information entropy, once all the information will be contained in the data set. If perfect training was achieved, error probability density would have the same shape as a Dirac impulse at the origin, and the estimated information particles probability density function would perfectly match the data set probability density function, this meaning that a perfect mapper would have been met.

Minimizing the error information entropy can be applied to perform supervised learning using a neural network for learning machine to implement the mapping function, and information entropy gradient back propagation algorithm. In fact, information entropy is estimated using the information potential function, which attains it's maximum when information entropy is at it's minimum. (2) states that if using neural networks for learning machines, one can back propagate information forces to maximize error entropy potential, therefore minimizing error entropy information.

Although, such solution has some known problems, like mean insensitivity, which degenerates in biased probability density function estimations, and growing quadratic complexity with respect to the number of data samples.

To diminish these problems influence on final results, correlation entropy has been presented in (4). In cited document, the definition of correlation entropy, also named correntropy, is presented and all it's properties outlined.

This information entropy criterion, as one may see in chapter 3, though in form very similar to the information potential, presents linear growth with respect to number of samples, like mean square error criterion does, and unbiased data set probability density function estimation.

If one uses correntropy for some supervised learning mechanism criterion, one will seek to maximize this function's value for the error data series. As depicted in chapter 2, such maximization has the probabilistic meaning of maximizing the error probability density function around the origin, thus tending to shape the error density distribution into a Dirac impulse shape centred at the origin. If desired Dirac shaping could be accomplished, an absolute zero error solution would have been found for all data samples. But to achieve such perfect training, one must surely wait for quantum based computer technology, and even though...

Nevertheless, perfect training doesn't have any real world application, since these real world applications do not possess perfect instruments, hence no perfect measure can be taken. Finite instrumentation resolution may well be used as a threshold for training objectives. Of course, some other measurement error quantifying function could be used like, for example, the measurement's uncertainty.

Training neural network mappers under the same conditions, using either error correntropy maximization criterion, either mean square error well known criterion, will be done so one can compare probability density functions and determine which of these methods performs best.

Now, let's assume that some mapper has been trained and correlation has been found between different signals. Well, under these conditions, this mapper should be able to restore it's missing signals based on the information contained in its parameters and known outputs (1). Obviously, the search of the best solution must be implemented through some optimization algorithm which will now move the outputs accordingly, instead of the mappers parameters.

Reminding that training criteria and learning machines are decoupled, any criterion may be adopted to search the best solution as long as it tends to minimize error information in some way.

In consequence, one will present a new algorithm to restore missing signals based on signal information correntropy and compare it with mean square error criterion, in terms of probabilistic quality of respectively estimated solutions.

Performance evaluation results are shown in chapters 4 and 5.

Comparison criteria comprises the number of iterations needed, shape of the error probability density function, error series maximum and minimum and respective mean square error.

Great part of the conclusions drawn are presented in these two chapters, but one may find its synthesis in chapter 6.

# Chapter 2

## Entropy and auto-encoders state of the art

## 2.1 Introduction

Information theoretic learning (2) offers an elegant mathematical approach to define a criterion which can be used to effectively minimize error information entropy.

Information theoretic learning tries to contour Gaussianity dependency of second order statistics to obtain an optimal solution.

To better understand error information entropy minimization criteria, a quick review and actual state of the art is here presented.

Entropy criterion advantages and disadvantages will be pointed out and discussed.

Correlation entropy, or simply, correntropy, is herein defined.

Auto-encoder state of the art is also discussed and presented as a possible solution to implement a neural network mapper which can be trained by means of a correntropy criterion.

## 2.2 Information theoretic learning state of the art

### 2.2.1 Information theoretic learning definition for information entropy

Information entropy first definitions date back to 1948, when Shannon (3) first defined a consensual definition for this pure quantity, which is associated to any kind of variable, independently of physical meaning:

$$H_s = \sum_{k=1}^{n} p_k I(p_k); \sum_{k=1}^{n} p_k = 1; p_k > 0$$

Other information entropy definition was presented by Renyi between 1960 and 1961:

$$H_{R\alpha} = \frac{1}{1-\alpha} \log\left( \sum_{k=1}^{n} p_k^{\alpha} \right); \alpha > 0; \alpha \neq 1$$

According to Kaplan (5), Renyi is an entropy maximum measure. Setting $\alpha$ equal to two, general Reniy's information entropy becomes Reniy's quadratic entropy:

$$H_{R2} = -\log\left( \sum_{k=1}^{n} p_k^2 \right)$$

This is a more convenient form, as one will see later.

Supposing a continuous random variable *X* with a probability density function $f_X(x)$, one can write the differential form for this equation:

$$H_{R\alpha} = \frac{1}{1-\alpha} \log\left( \int_{-\infty}^{+\infty} f_X(x)^{\alpha} dx \right); \alpha > 0; \alpha \neq 1$$

Xu discusses in (2) several non parametric methods to estimate the probability density function $f_X(x)$ , like histogram estimation, orthogonal series estimate and the one that has special interest for the remainder of the present document, Parzen window estimation.

Parzen window method, also known as kernel estimation method, or potential function method, estimates $f_X(x)$ by the use of some kernel based function. For latter use in this dissertation, a Gaussian kernel function is chosen to do such original variable estimation:

$$G(x,\sigma^2)=\frac{1}{(2\pi)^{\frac{k}{2}}\sigma^k}e^{-\left(\frac{x^T x}{2\sigma^2}\right)}$$

Which probability density function is:

$$f(x)=\frac{1}{N}\sum_{i=1}^{N}G\left(x-a(i),\sigma^2\right)$$

This in fact means that each data point will be substituted for a Gaussian estimation window which density is the average of all kernel functions.

Gaussian kernel is often chosen for Parzen estimator due to it's consistency in the various metrics making it a good estimate for a wide set of probability density classes and it's optimal to estimate smoother density distributions. Simplicity and good asymptotic properties finish the role of characteristics which favours Gaussian kernel to be chosen over others.

If some information about probability density distribution $f_X(x)$ is pre-known, one should select a Parzen window which better fits it.

Fisher (6) suggests an indirect way for entropy maximization, by means of mean squared error criterion:

$$J=\frac{1}{2}\int\left(u(x)-f_X(x)^2\,dx\right)$$

Where *u(x)* is a uniform pdf in a defined rectangular region *R* and $f_X(x)$ is the Parzen windowing estimate of the mapper's output pdf. A gradient method can be applied to minimize *J*, thus minimizing the quadratic error and, obviously, the output's mean square error.

These considerations are the necessary ones to understand information potential and information force concepts and make the connection between these concepts and supervised learning.

### 2.2.2 Quadratic entropy and information potential V

Integration of Shannon's entropy definition, even when applying Parzen window estimate, is an intractable problem. However, integration problems are considerably reduced, if one chooses the quadratic entropy definition. In this case, it is clear that when using Parzen window estimate, $H_{R\alpha}$ is equivalent to the convolution of two exponential functions.

Such exponential convolution is easily calculated as follows:

$$V=\int_{-\infty}^{+\infty} f_X(x)^2\,dx=\int_{-\infty}^{+\infty}\left(\frac{1}{N}\sum_{i=1}^{N}G(x-a(i),\sigma^2)\right)\left(\frac{1}{N}\sum_{j=1}^{N}G(x-a(j),\sigma^2)\right)\Leftrightarrow$$

$$\Leftrightarrow V=\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\int_{-\infty}^{+\infty}G(x-a(i),\sigma^2)G(x-a(j),\sigma^2)dx\Leftrightarrow$$

$$\Leftrightarrow V=\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}G(a(i)-a(j),2\sigma^2)$$

This function *V* is denominated information potential and determines the interaction between information particles. Correspondent Reniy quadratic entropy becomes:

$$H_{R2}=-\log(V)$$

Combining these two expressions it's possible to get detailed information about the distribution and structure of the data set.

### 2.2.3 Information forces *F*

Similarity between information potential *V* and potential energy *E* concept in physics, allows the introduction of information forces *IF* concept as the information potential *V* first derivative, equivalent to force *F* in mechanical systems, which is the potential energy *E* first derivative. This information driven force may move the interacting samples in order to change it's distribution and, consequently, the data set entropy.

Computing the information force *IF,* assuming an information quadratic potential *V* is quite straightforward:

$$IF_{a(i)}=\frac{dV}{da(i)}=-\left(\frac{1}{N^2\sigma^2}\right)\sum_{j=1}^{N}\sum_{j=1}^{N}G\left(a(i)-a(j),2\sigma^2\right)\left(a(i)-a(j)\right)$$

This expression can be interpreted as the summation of individual forces that particles *a(j)* impinge to particle *a(i)*.

In a mechanical system in which all particles can move freely in a given space, the potential energy may be changed by applying a given force to a particle, which will move towards the direction of the applied force. Likewise, information potential may be changed, either minimizing or maximizing it by moving information particles when applying an information force accordingly.

Although information entropy minimization or maximization is possible applying information forces to information particles, this solution does not guarantees an origin centred solution (7). In fact, Parzen window density estimation is a biased one and one must not forget that information forces minimize the euclidean distance between samples, and not between each sample and the origin, thus resulting it's mean insensitivity.

And let us not forget the heavy computational effort involved in some supervised learning

mechanism based in this information theoretic learning criterion. Such computational effort grows quadratically with the dimension of the data sets (8), and not linearly, like mean square entropy minimization.

These problems are actually solved by introducing the localized similarity measure baptized correlation entropy, or simply correntropy (9).

### 2.2.4 Generalized cross correlation entropy definition

Second order statistics, in particular, mean square error based, are probably the most widely implemented methodologies to quantify similarity between two random variables (4).

But, as stated before, optimal solution depends of some Gaussian statistical random variable behaviour, and many are the cases where such behaviour is not verifiable.

Higher order non-Gaussian signal processing requires higher order statistical functions to attain better solutions.

Let us then define a general form for cross correntropy $V_\sigma(x, y)$ as a generalized similarity measure between two arbitrary scalar random variables $x$ and $y$:

$$V_\sigma(x,y)=E[k_\sigma(x_i-y_i)]$$

This form of cross correntropy is estimated using Parzen windowing approximation. In this case, a Gaussian kernel $k_\sigma$ implements the Parzen window.

$$G(x,\sigma^2)=\frac{1}{(2\pi)^{\frac{1}{2}}\sigma}e^{-\left(\frac{x^T x}{2\sigma^2}\right)}$$

Once only a finite set of data is known, correntropy probability density function estimator is:

$$V_\sigma(x,y)=\frac{1}{N}\sum_{i=1}^{N}k_\sigma(x_i-y_i)$$

This cross correntropy function is also an higher order statistical similarity measure and at the same provides an algorithm which reduces computational effort to the same level of mean square error based supervised learning criterion, as one will see.

The properties related to cross correntropy metric are depicted in (4). In face of such properties, one can devise three different norm size regions for cross correntropy density estimator: when the two random variables are similar, it becomes an L2 norm, when they are very different, it becomes an L0 norm and in between these two, an L1 norm.

This norm size change explains the importance of the bandwidth $\sigma$ and cross correntropy density estimation robustness. Selecting a small bandwidth leads to a small euclidean zone search, yielding local optimal solutions, while large kernel sizes will increase the euclidean search region and this estimator behaves as the mean square error minimization method, yielding globally optimal solutions.

The most important interpretation one can make of such equation is that correntropy actually estimates the probability density of the equality between the random scalar variables *x* and *y*.

This local criterion of similarity is specially useful when signals with non-zero mean, non-Gaussian statistic behaviour and with large outliers are to be estimated (10).

### 2.2.5 Supervised learning under information entropy criteria

Supervised learning by means of some objective function intends to adjust a mapper *g(x, y)* output so that the error probability density function satisfies the objective function criterion.

Let us say that such mapper is a neural network mapper *g(I, w, b):*

$$O' = g(I, w, b)$$

So, the mapper outputs *O*' are the outputs of a function with respect to a set of inputs, *I*, a set of weights *w* and a set of neuron biases *b*.

Under supervised learning, one normally wishes to train such network in order to minimize the error information entropy. In fact, by minimizing the information contained in the error set, one maximizes the information contained in the data samples.

To bridge out to supervised learning under information theory, let us refer the most commonly used criterion to implement this error entropy minimization, the mean square error minimization method which objective function can be resumed to the minimization of the quadratic error:

$$\min_{w,b}\left(g(I,w,b)\right) = \min_{w,b}\left\{\frac{1}{2}\sum_{j=1}^{k}\left(O_j - g(I,w,b)\right)^2\right\} = \min_{w,b}\left\{\frac{1}{2}\sum_{j=1}^{k}\varepsilon_j^2\right\}$$

Surely, one can choose other criterion to perform such error entropy minimization due to known mean square error limitations.

In fact, let us rewrite the equation for the information potential substituting the two random variables for the respective error:

$$V = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} G(\varepsilon_i - \varepsilon_j, 2\sigma^2)$$

This expression represents the mutual variable error information potential. Maximizing the error information potential may well constitute the new training criterion (2):

$$\max_{w,b} V = \max_{w,b}\left(\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} G(\varepsilon_i - \varepsilon_j, 2\sigma^2)\right)$$

But refered problems, lead to the development of the correntropy definition:

$$V_\sigma(\varepsilon_i, \varepsilon_j) = \frac{1}{N}\sum_{i=1}^{N} k_\sigma(\varepsilon_i - \varepsilon_j)$$

This is a much simpler approach to the same problem: minimizing error information entropy by maximizing it's potential. But this equation as is would output biased probability density function, once it's formulation is very similar to the definition of information potential *V*. According to (4) one can compare a random variable to a given fixed point without loosing any of it's intrinsic properties and metric.

Thus, if one desires to minimize the error information entropy by maximizing error correntropy around the origin, one may then write:

$$V_\sigma(0,\varepsilon_j)=\frac{1}{N}\sum_{j=1}^{N}k_\sigma(0-\varepsilon_j)=\frac{1}{N}\sum_{j=1}^{N}k_\sigma(g(I,w,b)-O_j)$$

Where $O_j$ is the targeted mapper output.

Finally, the correntropy based objective function is:

$$\max_{w,b}\left(J(w,b)\right)=\max_{w,b}\left(V_\sigma(\varepsilon)\right)=\max_{w,b}\left(\frac{1}{N}\sum_{j=1}^{N}k_\sigma(-\varepsilon_j)\right)$$

The maximization of this objective function is equivalent to maximizing the error probability density distribution of the random scalar variable *x* around the origin and consequently minimizing error entropy.

Desired output is a Dirac shaped error probability density centred at the origin. Of course, practical issues impose limitations to obtain such optimal solution in which all errors would be absolute zero.

Hence, some more reasonable criteria, like ensuring all error particles fall inside a given bandwidth, which limits could be, for example, the maximum measurement resolution is normally adopted.

Although, one might try and reach for perfection...

## 2.3 Auto-encoders state of the art

Auto-encoders, also known as auto-associative mappers, aren't more than neural networks with the particularity of mirroring in its outputs the respective inputs, as shown in the following figure 3.

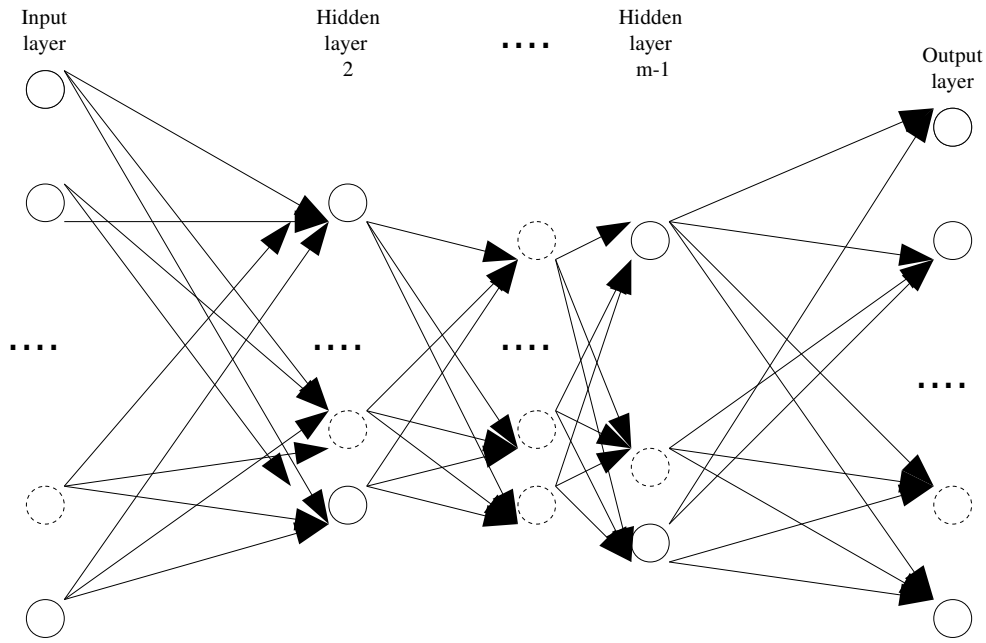


**Figure 3: Multi-layer auto-encoder.**

Theoretically speaking, feed forward neural networks state of the art has not seen any development since Rumelhart et al. (11) defined multi-layer perceptron theory and introduced the concept of artificial neurons, which activation functions are continuously differentiable, thus permitting supervised learning by the application of objective function gradient search through neural network back propagation.

In practice, some recent investigations lead to the conclusion that auto-encoders are an excellent base structure to encode signals, hence, able to establish some correlation between apparently chaotic and uncorrelated data samples.

As one may see in subsequent chapter 5, auto-encoder properties, which include it's signal restoration abilities, made us choose this type of neural network to implement the mapper needed to encode and restore missing signals.

Information compression in a single hidden layer auto-encoder is one of the objectives of the present dissertation. Therefore, only single hidden layer auto-encoders will be studied then.

To concede maximum information compression, this intermediate layer must have the minimum number of neurons possible and no neuron bias will be considered neither adjusted through the remainder of this document.

More will follow about objective function back propagation and auto-encoders in the succeeding chapters.

# Chapter 3

## Back propagation training by means of an entropy criterion

## 3.1 Introduction

As refered before, for an auto-encoder to perform as expected some kind of supervised learning algorithm must be applied to do so.

The feed forward neural network training algorithm most commonly found in literature is based in the well known mean square error minimization method.

In consequence of the feed forward neural network that an auto-encoder is, the mean square error minimization algorithm is back propagation based.

Let us remember the basics about this training algorithm when applied to feed forward neural networks, starting with some definitions and how forward propagation works.

## 3.2 Feed forward neural network topology forward propagation

Defining $I$ as the input values vector, $O$ as the desired output values vector, $N_{ij}$ the $j^{th}$ neuron output value of the $i^{th}$ layer, being the first one the input and the last one the output layer, and finally $w_{ij}^{k}$ the weight of the synapse that connects the neuron $i$ of the layer $(k-1)$ to the neuron $j$ of the layer $k$. Assuming that associated to each neuron there is a transfer function $\Phi(I)$ the respective output $N_{ij}$ will be the following:

$$N_{ij}=\Phi(I)=\Phi(I_{ij}+b_{ij})$$

Where $I_{ij}$ is the input of the $j^{th}$ neuron in the $i^{th}$ layer and $b_{ij}$ the corresponding bias.

For the input layer, it is straightforward to deduce that:

$$I_{ij}=I_{j}$$

For the remaining layers, and taking in consideration the following artificial neuron model:

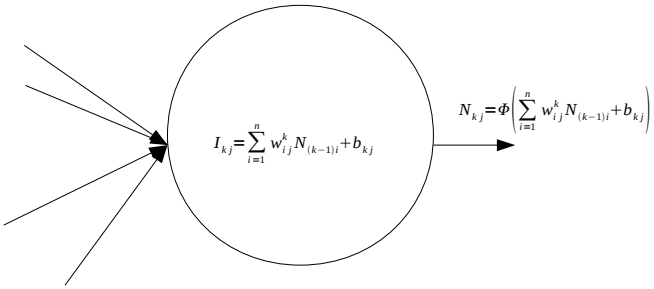


**Figure 4: Artificial neuron model**

One can deduce that for the neuron $j$ of the $k^{th}$ layer, the input will be:

$$I_{kj}=\sum_{i=1}^{n} w_{ij}^{k} N_{(k-1)i}+b_{kj}$$

And the output for the given neuron is:

$$N_{kj}=\Phi\left(\sum_{i=1}^{n} w_{ij}^{k} N_{(k-1)i}+b_{kj}\right)$$

This forward propagation is done successively layer by layer stoping only once the outputs of the neurons composing the output layer are calculated. This resumes the feed forward propagation mechanism used to propagate the input signals across the neural network which generate the output set of values *O'*. *A*ccording to the chosen nomenclature, the $j^{th}$ output neuron of an *m* layer sized network corresponds to:

$$O'_j = N_{mj}$$

If no training has been done and admiting some arbitrary initial values have been attributed to neuron biases $b_{ij}$ and synapses weights $w_{ij}^{k}$, discrepancy between the desired output values *O* and the output $O'$ obtained by means of forward propagation will surely verify. Let us then define the absolute error vector $\varepsilon$ :

$$\varepsilon = O - O'$$

## 3.3 Mean square error minimization criterion under back propagation training

Now, one may consider the vector $\varepsilon$ as the measure used by the mean square error minimization algorithm to train the network by means of back propagation successive iterations, once this process may be repeated while particularly defined conditions aren't met like, for example, finding a solution where the outputs mean square error is smaller than a given threshold.

In more strict terms, the error back propagation attempts to minimize the mean square error for a zero threshold:

$$\min_{w,b}\left(H(w,b)\right)=\min_{w,b}\left\{\frac{1}{2}\sum_{j=1}^{k}\left(O_j-g(I,w,b)\right)^2\right\}=\min_{w,b}\left\{\frac{1}{2}\sum_{j=1}^{k}\left(O_j-O'_j\right)^2\right\}\Leftrightarrow$$

$$\Leftrightarrow\min_{w,b}\left(H(w,b)\right)=\min_{w,b}\left\{\frac{1}{2}\sum_{j=1}^{k}\varepsilon_j^2\right\}$$

But, computational effort, discrete computer variables and associated mantissa errors, added to other practical issues, generally obligates us to set a non-zero threshold.

Such mean square error minimization is achieved by adjusting neuron biases $b_{ij}$ and networks synapses weights $w_{ij}^{k}$ .

Let us review the mean square error back propagation algorithm bridging to what is exposed throughout the remaining chapters of this document.

Once the absolute error has been calculated and stored in $\varepsilon$ it is possible to adjust the neuron biases $b_{ij}$ and synapses weights $w_{ij}^{k}$ using an error gradient descent technique.

Starting the back propagation at the output layer, to calculate the contribution of the neuron bias to the error expressed by the $j^{th}$ output neuron:

$$\Delta b_{mj} = -\left(\frac{dH(w,b)}{db_{mj}}\right) = -\left(\frac{dH(w,b)}{dO'_j}\frac{dO'_j}{dI_{mj}}\frac{dI_{mj}}{db_{mj}}\right)$$

The adjustment calculation for each synapse weight is done in the same way:

$$\Delta w_{ij}^{m} = -\left(\frac{dH(w,b)}{dw_{ij}^{k}}\right) = -\left(\frac{dH(w,b)}{dO'_j}\frac{dO'_j}{dI_{mj}}\frac{dI_{mj}}{dw_{ij}^{k}}\right)$$

This last adjustment must be calculated for every synapse that connects the present neuron to any neuron in the preceding hidden layer *(m-1)*.

Assuming a continuously differentiable neuron activation function $\Phi(x)$ :

$$\Delta w_{ij}^{m} = \varepsilon_j \Phi'(I_{mj})\frac{dI_{mj}}{dw_{ij}^{k}} \qquad \Delta b_{mj} = \varepsilon_j \Phi'(I_{mj})\frac{dI_{mj}}{db_{mj}}$$

Where $\Phi'(x)$ is function $\Phi(x)$ derivative and $I_{mj}$ is the present neuron input.

To calculate the last derivative, summation is neglected, once this calculation is done with respect to the present neuron:

$$\Delta w_{ij}^{m} = \varepsilon_j \Phi'(I_{mj}) N_{(m-1)i}$$
$$\Delta b_{mj} = \varepsilon_j \Phi'(I_{mj})$$

Before setting the synapses corrected values, it's necessary to back propagate the error through the network.

The contribution of the present layer neuron input error once propagated backward generates at the output of the preceding layer neurons to which it's connected to:

$$N_{(m-1)i} = w_{ij}^{m} \Delta w_{ij}^{m}$$

For each neuron of the preceding layer *(m-1)*, it is necessary to sum up the error contribution of all the neurons to which it's connected to in the $m^{th}$ layer:

$$N_{(m-1)i} = \sum_{j=1}^{k} \left(w_{ij}^{m} \Delta b_{mj}\right)$$

The new synapse weights $w_{ij}^{m}{}'$ connecting layer *(m-1)* to the $m^{th}$ layer and the $m^{th}$ layer neuron biases $b_{mj}{}'$ will be:

$$b_{mj}{}' = b_{mj} + \psi \Delta b_{mj}$$
$$w_{ij}^{m}{}' = w_{ij}^{m} + \psi \Delta w_{ij}^{m}$$

Where $\psi$ is the learning rate for each iteration:

$$\psi > 0$$

This procedure is repeated until the new input neuron biases are calculated and respective connections weight are set, thus terminating one complete training iteration.

Suppose now that there are *n* input patterns and respective output patterns.

For each pattern *z*, the training method will be the same as the one already depicted, except if training is done in batch mode. In this case, no changes are made to networks synapse weights and neuron biases until all pattern error contributions are summed up, resulting for the adjusted neuron biases and synapse weights at the end of an iteration, that is, the complete pattern set:

$$b_{mj}{}' = b_{mj} + \psi \sum_{z=1}^{n} (\Delta b_{mj})_z$$
$$w_{ij}^{m}{}' = w_{ij}^{m} + \psi \sum_{z=1}^{n} (\Delta w_{ij}^{m})_z$$

Once understood the mean square error minimization training algorithm, one can easily consider other ways to measure the difference between forward propagation output *O'* and desired output values *O*, but in what way are these solutions equally able to adjust biases and synapses weights?

In fact, one may choose some other already proven valid criterion to train some neural network, such as evolutionary computing methods, like particle swarm optimization, or simpler straightforward derived methods from optimization theory, like the one here described.

Although, let us not mistake simplicity for lack of efficiency.

## 3.4 Error correlation entropy maximization criterion under back propagation training

Once reviewed the mean square error minimization criterion applied to supervised neural network learning by means of back propagation, it is quite straightforward to delineate an adaptation which complies with error correlation entropy, or simply, correntropy, maximization criterion.

As one will see, no added complexity is needed to perform neural network back propagation training when adopting this new criterion.

The criterion now adopted, according to what has already been defined in preceding chapter 2, is to maximize error correntropy:

$$\max_{w,b}\left(J(w,b)\right)=\max_{w,b}\left[\frac{1}{N}\sum_{j=1}^{N}k_{\sigma}\left(g(I,w,b)-O_j\right)\right]=\max_{w,b}\left[\frac{1}{N}\sum_{j=1}^{N}k_{\sigma}\left(O'_j-O_j\right)\right]\Leftrightarrow$$
$$\Leftrightarrow\max_{w,b}\left(J(w,b)\right)=\max_{w,b}\left[\frac{1}{N}\sum_{j=1}^{N}k_{\sigma}\left(-\varepsilon_j\right)\right]$$

Remind that a function maximization corresponds to it's symmetric minimization:

$$\max_{w,b}\left(J(w,b)\right)=\min_{w,b}\left(-J(w,b)\right)=\min_{w,b}\left(Q(w,b)\right)$$

Applying the depicted back propagation gradient descent method for the maximization of the error's information correntropy, weights composing the neural network mapper $w_{ij}^{k}$ must be adjusted in accordance to the gradient associated to this objective function. The same proceeding used to minimize the error through back propagation may now be applied considering the new objective function:

$$\frac{dQ(w_{ij}^{m})}{dw_{ij}^{m}}=-\left(\frac{dJ(w_{ij}^{m})}{dw_{ij}^{m}}\right)=-\left(\frac{dJ(w_{ij}^{m})}{dO'_j}\frac{dO'_j}{dw_{ij}^{m}}\right)$$

Once this is done neuron by neuron, the summation is omited, thus remaining only the term refering to the specific neuron, let us say, *j*. In accordance to what has been defined in chapter 2:

$$k_{\sigma}\left(O'_j-O_j\right)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\left(\frac{(O'_j-O_j)^2}{2\sigma^2}\right)}$$

Calculating this equation derivative with respect to the output $O'_j$, results in:

$$\frac{dQ(w_{ij}^{m})}{dw_{ij}^{m}}=\left(\frac{O'_j-O_j}{\sqrt{2\pi}\sigma^3}\right)e^{-\left(\frac{(O'_j-O_j)^2}{2\sigma^2}\right)}\frac{dO'_j}{dw_{ij}^{m}}$$

Further decomposing this equation to obtain the $j^{th}$ neuron input contribution, once its output is its activation function output value, evaluated at its input value $I_{mj}$ :

$$O'_j=\Phi(I_{mj})$$

Assuming that $\Phi(x)$ is a continuously differentiable function, one can write:

$$\frac{dQ(w_{ij}^{m})}{dw_{ij}^{m}}=\left(\frac{O'_j-O_j}{\sqrt{2\pi}\sigma^3}\right)e^{-\left(\frac{(O'_j-O_j)^2}{2\sigma^2}\right)}\frac{dO'_j}{dI_{mj}}\frac{dI_{mj}}{dw_{ij}^{m}}$$

Therefore:

$$\frac{dQ(w_{ij}^m)}{d\,w_{ij}^m}=\left(\frac{O'_j-O_j}{\sqrt{2\pi}\,\sigma^3}\right)e^{-\left(\frac{(O'_j-O_j)^2}{2\sigma^2}\right)}\Phi'(I_{mj})\frac{dI_{mj}}{dw_{ij}^m}$$

For the last term, it is known that for the $j^{th}$ neuron input $I_{mj}$ we have:

$$I_{mj}=\sum_{i=1}^{n} w_{ij}^m N_{(m-1)i}+b_{mj}$$

Once the calculation considers the summation's single term which contemplates the specific synapse weight connecting the $i^{th}$ neuron of the *(m-1)* layer to the $j^{th}$ output neuron in the $m^{th}$ layer, the expression will become:

$$\frac{dQ(w_{ij}^m)}{d\,w_{ij}^m}=\left(\frac{O'_j-O_j}{\sqrt{2\pi}\,\sigma^3}\right)e^{-\left(\frac{(O'_j-O_j)^2}{2\sigma^2}\right)}\Phi'(I_{mj})N_{(m-1)i}$$

The synapse weight adjustment $\Delta w_{ij}^m$ will finally be:

$$\Delta w_{ij}^m=-\left(\frac{dQ(w_{ij}^m)}{d\,w_{ij}^m}\right)=\left(\frac{O_j-O'_j}{\sqrt{2\pi}\,\sigma^3}\right)e^{-\left(\frac{(O_j-O'_j)^2}{2\sigma^2}\right)}\Phi'(I_{mj})\,N_{(m-1)i}$$

Defining $\gamma$ as the learning rate:

$$\gamma>0$$

The adjusted synapse weight $w'^m_{ij}$ will then be:

$$w'^m_{ij}=w_{ij}^m+\gamma\,\Delta w_{ij}^m=w_{ij}^m-\gamma\,\frac{dQ(w_{ij}^m)}{d\,w_{ij}^m}=w_{ij}^m+\gamma\left(\frac{\varepsilon_j}{\sqrt{2\pi}\,\sigma^3}\right)e^{-\left(\frac{\varepsilon_j^2}{2\sigma^2}\right)}\Phi'(I_{mj})N_{(m-1)i}$$

Surely, this process must be repeated for each output neuron and respective hidden layer connecting synapses.

Preceding such weight correction, one must back propagate this neuron's contribution for the hidden layer neuron output, $N'_{(m-1)i}$, to which is connected to through synapse $w_{ij}^m$ :

$$N'_{(m-1)i}=w_{ij}^m\left(\frac{O_j-O'_j}{\sqrt{2\pi}\,\sigma^3}\right)e^{-\left(\frac{(O_j-O'_j)^2}{2\sigma^2}\right)}\Phi'(I_{mj})$$

Obviously, it is necessary to find the total contribution for the hidden layer neuron output $N'_{('m-1)\,i}$, made by each of the neurons $o$ of the output layer to which is connected to:

$$N'_{(m-1)i} = \sum_{j=1}^{o} w_{ij}^{m} \left( \frac{O_j - O'_j}{\sqrt{2\pi}\sigma^3} \right) e^{-\left( \frac{(O_j - O'_j)^2}{2\sigma^2} \right)} \Phi'(I_{mj})$$

Repeating the process here already depicted, and again supposing a continuously differentiable hidden layer neurons activation function $\Phi(x)$, for the synapse connecting the hidden layer neuron $N_{(m-1)\,i}$ to the hidden layer neuron $N_{(m-2)\,z}$, the weight adjustment $\Delta w_{zi}^{m-1}$ is:

$$\Delta w_{zi}^{m-1} = N'_{(m-1)i} \Phi'(I_{(m-1)i}) N_{(m-2)\,z}$$

Once more, before doing this weight update, back propagation of this neuron *i* input contribution to the preceding hidden layer neuron output must be computed:

$$N'_{(m-2)z} = w_{zi}^{m-1} N'_{(m-1)i} \Phi'(I_{(m-1)i})$$

Finally, the adjustment of the synapses weights is:

$$w'^{m-1}_{zi} = w_{zi}^{m-1} + \gamma \Delta w_{zi}^{m-1}$$

This procedure is to be repeated until the complete set of weights of the synapses connecting the input neurons $N_{0\,r}$ to the first hidden layer neurons $N_{1t}$ are calculated:

$$w'^{1}_{rt} = w_{rt}^{1} + \gamma \Delta w_{rt}^{1}$$

At the same time, the algorithm sets the new neuron biases using the same objective function, but now with respect to the neuron biases:

$$\frac{dQ(b_{mj})}{db_{mj}} = -\left( \frac{dJ(b_{mj})}{db_{mj}} \right) = -\left( \frac{dJ(b_{mj})}{dO'_j} \frac{dO'_j}{db_{mj}} \right)$$

Applying the chain rule, this expression becomes:

$$\frac{dQ(b_{mj})}{db_{mj}} = \left( \frac{O'_j - O_j}{\sqrt{2\pi}\sigma^3} \right) e^{-\left( \frac{(O'_j - O_j)^2}{2\sigma^2} \right)} \frac{dO'_j}{dI_{mj}} \frac{dI_{mj}}{db_{mj}}$$

Once $\Phi(x)$ is assumed again to be the continuously differentiable output neuron activation function:

$$\frac{dQ(b_{mj})}{db_{mj}} = \left( \frac{O'_j - O_j}{\sqrt{2\pi}\sigma^3} \right) e^{-\left( \frac{(O'_j - O_j)^2}{2\sigma^2} \right)} \Phi'(I_{mj}) \frac{dI_{mj}}{db_{mj}}$$

Once this calculation is done for the neuron bias $b_{mj}$ only:

$$\Delta b_{mj} = -\left(\frac{dQ(b_{mj})}{d\,b_{mj}}\right) = \left(\frac{O_j - O'_j}{\sqrt{2\pi\sigma^3}}\right) e^{-\left(\frac{(O_j - O'_j)^2}{2\sigma^2}\right)} \Phi'(I_{mj})$$

The new bias $b'_{mj}$ will be:

$$b'_{mj} = b_{mj} + \gamma\,\Delta b_{mj} = b_{mj} + \gamma\left(\frac{\varepsilon_j}{\sqrt{2\pi\sigma^3}}\right) e^{-\left(\frac{\varepsilon_j^2}{2\sigma^2}\right)} \Phi'(I_{mj})$$

For the preceding hidden layer, the neuron bias *i* is to be adjusted by an amount which is proportional to the back propagated error contributions made by each neuron of the output layer to which it's connected to:

$$\Delta b_{(m-1)i} = N'_{(m-1)i} \Phi'(I_{(m-1)i})$$

The adjusted bias will be set to:

$$b'_{(m-1)i} = b_{(m-1)i} + \gamma\,\Delta b_{(m-1)i}$$

This is done until all neuron biases are adjusted, obviously finishing at the input layer:

$$b'_{1r} = b_{1r} + \gamma\,\Delta b_{1r}$$

This finishes one iteration of the back propagation algorithm using error correntropy maximization criterion.

Obviously, one can repeat this process to pursue better solutions.

Let us now suppose one wishes to train a neural network using *n* input patterns and respective output patterns. For each pattern *z*, the training method will be the same as the one already depicted, except if training is done in batch mode. In this case, no changes are made to networks synapse weights and neuron biases until all pattern error contributions are summed up, resulting for the adjusted neuron biases and synapse weights at the end of an iteration, that is, for the complete pattern set:

$$w'^{m}_{ij} = w^{m}_{ij} + \gamma \sum_{z=1}^{n} \left(\Delta w^{m}_{ij}\right)_z$$

$$b'_{mj} = b_{mj} + \gamma \sum_{z=1}^{n} \left(\Delta b_{mj}\right)_z$$

Naturally, this will be done for all neural network neuron biases and synapses.

# Chapter 4

## Training auto-encoders by means of an entropy criterion

## 4.1 Introduction

Derived from what has been described in previous chapters, training auto-encoders by means of a correntropy criterion is theoretically quite straightforward.

Reminding that this study is focused on single hidden layer auto-encoders so that one can evaluate its hidden layer data compression abilities, the back propagation mathematics refering to correntropy information criterion comprising solely 3 layers, namely, input, hidden or compression layer, and output layer, becomes trivial.

Let us not forget that no neuron bias adjustment will be done during training sessions. Neuron biases will in fact work as mapping constraints, so one can evaluate how do training algorithms are able contour “bumps” and “holes” left by neuron biases impression.

Assume an auto-encoder topology according to the one shown in figure 1.

Naming $I$ as the input vector and its equivalent desired output vector $O$, $O'$ as the neural network forward propagation output outcome of its mapping function:

$$O' = g(I, w)$$

And resulting error vector $\varepsilon$:

$$\varepsilon \equiv O - O' \equiv I - O'$$

Applying the optimal solution definition for correntropy implementing Parzen windowing estimation as stated in (4) :

$$\max_{w} J(w) = \frac{1}{N}\sum_{j=1}^{N} k_{\sigma}\left(g(I, w) - O_j\right) = \frac{1}{N}\sum_{j=1}^{N} k_{\sigma}\left(O'_j - O_j\right) = \frac{1}{N}\sum_{j=1}^{N} k_{\sigma}\left(-\varepsilon_j\right)$$

This expression has the probabilistic meaning of maximizing the error density function around the origin (4) and at the limit:

$$\varepsilon = 0 \Rightarrow O' \equiv I \equiv O \equiv g(I_i, w)$$

In this case, a perfect auto-encoder mapping function $g(I, w)$ would have been met.

## 4.2 Auto-encoder back propagation brief review

Briefly, based on preceding chapter reviewed back propagation gradient descent method for the maximization of the error's information correntropy, one must move through weight space proportionally adjusting each weight $w_{ij}^{k}$ in accordance to the gradient associated to the objective function. Applying the derivation chain rule we will get:

$$\frac{dJ(w_{ij}^{k})}{d\,w_{ij}^{k}}=\frac{dJ(w_{ij}^{k})}{dO'_{j}}\frac{dO'_{j}}{d\,w_{ij}^{k}}$$

Since we are calculating this gradient for a single output neuron *j*, the summation is omited and considered only the respective term. Once:

$$k_{\sigma}\left(O'_{j}-O_{j}\right)=\frac{1}{\sqrt{2\pi}\,\sigma}\,e^{-\left(\frac{(O'_{j}-O_{j})^{2}}{2\sigma^{2}}\right)}$$

The, expression above will now be:

$$\frac{dJ(w_{ij}^{k})}{d\,w_{ij}^{k}}=-\left(\frac{O'_{j}-O_{j}}{\sqrt{2\pi}\,\sigma^{3}}\right)e^{-\left(\frac{(O'_{j}-O_{j})^{2}}{2\sigma^{2}}\right)}\frac{dO'_{j}}{d\,w_{ij}^{k}}$$

This equation's right term can still be decomposed to obtain the j[th] neuron input contribution, once its output is its activation function output value, evaluated at its input value $I_{kj}$ :

$$O'_{j}=\psi(I_{kj})$$

Assuming that $\psi(x)$ is a continuously differentiable function, one can write:

$$\frac{dJ(w_{ij}^{k})}{d\,w_{ij}^{k}}=-\left(\frac{O'_{j}-O_{j}}{\sqrt{2\pi}\,\sigma^{3}}\right)e^{-\left(\frac{(O'_{j}-O_{j})^{2}}{2\sigma^{2}}\right)}\frac{dO'_{j}}{dI_{kj}}\frac{dI_{kj}}{dw_{ij}^{k}}$$

Therefore:

$$\frac{dJ(w_{ij}^{k})}{d\,w_{ij}^{k}}=-\left(\frac{O'_{j}-O_{j}}{\sqrt{2\pi}\,\sigma^{3}}\right)e^{-\left(\frac{(O'_{j}-O_{j})^{2}}{2\sigma^{2}}\right)}\psi'(I_{kj})\frac{dI_{kj}}{dw_{ij}^{k}}$$

For the last term, it is known that for the j[th] neuron input $I_{kj}$ we have:

$$I_{kj}=\sum_{i=1}^{n}w_{ij}^{k}N_{(k-1)i}+b_{kj}$$

Once the calculation considers the summation's single term which contemplates the specific synapse weight connecting the $i^{th}$ neuron of the *(k-1)* layer to the $j^{th}$ output neuron in the $k^{th}$ layer, the expression will become:

$$\frac{dJ(w_{ij}^k)}{d\,w_{ij}^k} = -\left(\frac{O'_j - O_j}{\sqrt{2\pi}\,\sigma^3}\right) e^{-\left(\frac{(O'_j - O_j)^2}{2\sigma^2}\right)} \psi'(I_{kj})\, w_{ij}^k$$

The synapse weight adjustment $\Delta w_{ij}^k$ will finally be:

$$\Delta w_{ij}^k = -\gamma\left(\frac{-dJ(w_{ij}^k)}{d\,w_{ij}^k}\right) = \gamma\left(\frac{O_j - O'_j}{\sqrt{2\pi}\,\sigma^3}\right) e^{-\left(\frac{(O_j - O'_j)^2}{2\sigma^2}\right)} \psi'(I_{kj})\, w_{ij}^k$$

Where $\gamma$ is the learning rate. Finally, the adjusted synapse weight $w'^k_{ij}$ will then be:

$$w'^k_{ij} = w_{ij}^k + \Delta w_{ij}^k = w_{ij}^k - \gamma\left(\frac{dJ(w_{ij}^k)}{d\,w_{ij}^k}\right) = w_{ij}^k\left[1 + \gamma\left(\frac{O_j - O'_j}{\sqrt{2\pi}\,\sigma^3}\right) e^{-\left(\frac{(O_j - O'_j)^2}{2\sigma^2}\right)} \psi'(I_{kj})\right]$$

Surely, this process must be repeated until all output neurons and respective hidden layer connecting synapses are adjusted.

Preceding such weight correction, one must back propagate this neuron's contribution for the hidden layer neuron output, $H_i$, to which is connected to through synapse $w_{ij}^k$ :

$$H_i = w_{ij}^k \Delta w_{ij}^k = \gamma\left(\frac{O_j - O'_j}{\sqrt{2\pi}\,\sigma^3}\right) e^{-\left(\frac{(O_j - O'_j)^2}{2\sigma^2}\right)} \psi'(I_{kj}) \left(w_{ij}^k\right)^2$$

Obviously, it is necessary to find the total contribution for the hidden layer neuron output $H_i$, made by each of the neurons of the output layer to which is connected to:

$$H_i = \sum_{j=1}^{o} w_{ij}^k \Delta w_{ij}^k$$

Repeating the process here already depicted, and again supposing a continuously differentiable hidden layer neurons activation function $\psi(x)$ , for the synapse connecting the hidden layer neuron $H_i$ to the input layer neuron $J_z$ , the weight adjustment $\Delta w_{zi}^{k-1}$ is:

$$\Delta w_{zi}^{k-1} = H_i \psi'(I_{(k-1)i})\, w_{zi}^{k-1}$$

The training process will be over once all weights have been updated:

$$w'^{k-1}_{zi} = w_{zi}^{k-1} + \Delta w_{zi}^{k-1}$$

## 4.3 Problem Presentation

We have been given access to data refering to an eolic power central sensor readings comprising a 9 month period registered with a constant 10 minutes sampling rate. This data gathers information about the wind speed, $S$, wind direction, $D$, total eolic central real, $P'$, and reactive, $R$, electric power outputs, and 12 individual eolic towers real electric power output, $P_i$.

At first glance, the data collected appears to have a chaotic behaviour and no apparent correlation between samples is observable, as one can demonstrate using the following graphic, which depicts the wind direction sine as a function of the total real electric power output for a 6 days time window.

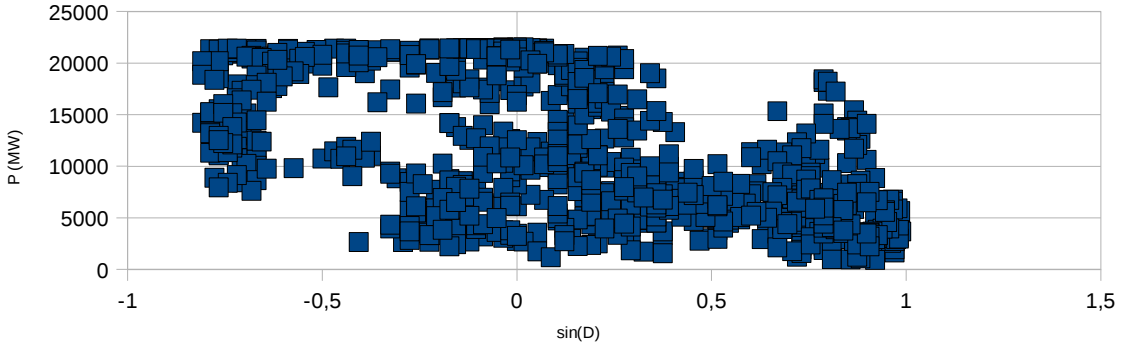


**Figure 5: Wind direction sine and total real electric power output graphic for a 6 days period.**

The problem can be summarized to identify any possible correlation between simultaneously collected readings and, moreover, if it's possible to establish some correlation between readings obtained at consecutive time instants.

If this intermediate goal is achieved, it will allow us to use these trained auto-encoders to help us solve missing signal restoration problems.

Auto-encoder training will be done using the mean squared error minimization back propagation method, the control method, and error information entropy maximization back propagation method.

The objective is to study and compare performances between trained minimum error entropy and minimum mean square error auto-encoders.

## 4.4 Problem formulation

To establish some correlation between simultaneously and time separated eolic power central acquired information, one may recur to neural networks, specifically, auto-encoders, as shown in previous chapters.

But some definitions are in order to better expose what was done for this particular case study.

As previously stated, an auto-encoder identifies the existence of correlation within a given set of information particles when able to define a mapping function $g(I,w)$ that ideally sets the neural network outputs $O'$ equal to its inputs $I$:

$$O' = g(I,w) = I$$

Let us define the minimum size input vector $S_t$, composed by one complete set of simultaneously acquired information particles occurred at a given time instant $t$:

$$S_t \equiv \{P'_t, R_t, P_{1t}, \ldots, P_{12t}, S_t, D_t\}^T$$

Defining the input vector $I_S$ as the concatenation of a fixed number $n$ of these vectors:

$$I_s \equiv \{P'_1, R_1, P_{11}, \ldots, P_{121}, S_1, D_1, \ldots, P'_n, R_n, \ldots, P_{12n}, S_n, D_n\}^T$$

By corresponding each input sample to an input neuron, each neuron can be seen as a sensor reading at a given time instant. To better picture what is here defined, refer to figure 1.

This particular way to encode the sensor reading time series allows the auto-encoder to find any possible correlation that may underlie between sets of samples, $S_t$, if training results consubstantiate such correlation.

The auto-encoder input and output layers have the same size as this input vector $I_s$. By definition, this vector $I_S$ is equivalent to the targeted output vector $O_S$, for an auto-encoding point of view. This vector is also the reference for the auto-encoder supervised learning applied algorithms:

$$I_s \equiv O_S$$

Due to hardware limitations, the maximum size for the auto-encoders input and output used for simulation was set to be 512 neurons, equivalent to 32 consecutively read sets $S_t$, covering a 6h period of time.

The trained auto-encoder should correctly encode a 6 days journey, this meaning that it must satisfy simultaneously 25 input sets, approximately.

The training will be done for each input set $I_s$ as explained in precedent chapter, in arbitrary order, if one wishes so. On the other hand, to ensure data connectivity between consecutive pattern inputs $I_S$, it was set an approximately 20% sample redundancy between consecutive inputs $I_S$, this meaning that the last 6 sample sets $S_t$ of a given input vector $I_S$ are the next input vector $I_{S+1}$ first 6 sample sets.

The stopping criterion adopted stops training as soon as one of these conditions is met: an absolute error below sensor reading's resolution verifies for all readings or iteration 80000 has been reached.

## 4.5 Simulations

The hidden layer size was set to 51 neurons, after some unsatisfying rehearsals with smaller sized ones.

The signal's span forced us to compress it to a smaller sized space *G*, bounding all the signals *s* composing $S_t$ to:

$$-1 \leqslant s \leqslant 1$$

In the absence of such signal normalization, tested auto-encoders weren't able to perform as wished for either supervised learning algorithms. It's a fact that the amount of data involved, the chosen network topology, which imposes combinatorial limitations, allied to the chosen hyperbolic tangent neuron activation function, which compresses the neuron's input signals to an output space bounded to the same interval defined for the signals *s*, reduces the possible solution space span for the neural network mapper $g(I, w)$ .

Initial weights $W_{ij}^{k}$ were randomly set to values comprised in the interval $[-0.1, 0.1]$ .

To verify training algorithms repeatability, 10 different initial weight sets were used to find a convenient mapper for the same input set. On the other hand, 10 different input sets were mapped by 10 different initial weight sets. The consecutiveness of all input sets was done with the clear purpose of covering a 60 consecutive days period. Now, if one considers the individual auto-encoder mappers part of a bigger map $G(I, w)$ , it is clear that the correspondent super neural network will comprise all the information for the 60 days period.

To facilitate performance comparison, the initial conditions were the same for both training algorithms.

Training under information correntropy maximization criterion was realized applying successively smaller sized Parzen window kernels, to ensure better training results (2). Such adjustment was initially done obeying Silverman's rule exposed in (12). Once the training's outcome using this rule didn't satisfy defined success conditions, kernel size was then set by scanning method. Successive kernel adjustments were done only when a local optimal solution was found, that is, whenever training iterations overall resulted in a constant output for the objective function.

The gain parameter was also continuously reduced as an attempt to maintain objective function output scaled to reasonable numeric values.

The stage is now set to show the results obtained under the specified terms.

For the sake of simplicity, only the global per-sensor results are here displayed.

Consequently, the chosen sets used to obtain the printed histograms concatenate the output errors vector obtained by several auto-encoders mapping functions. These error vectors, as already described, store the per-sensor output errors for 10 different auto-encoders trained to properly auto-encode the same training pattern, these ones displayed first (figures 6 to 37), followed by the histograms for 10 auto-encoders, each one trained to auto-encode one of 10 possible sample sets (figures 38 to 69).

Obviously, output error probability distribution functions were drawn for both pointed supervised learning methods and to allow better comparison these are displayed side by side, grouped by sensor. Notice that these histograms aren't graphically comparable. To distinguish the diamonds from the glass pieces one must not forget to check graphic scales.

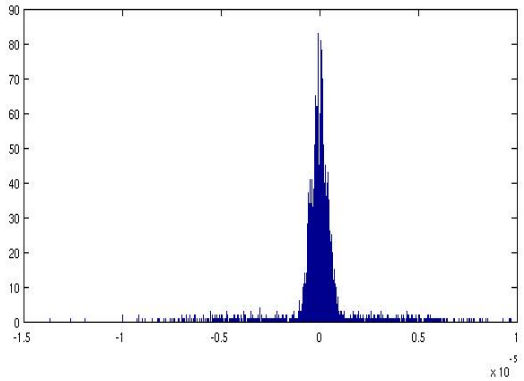


Figure 6: Max-Correntropy error pdf for a single pattern. sensor *P'*

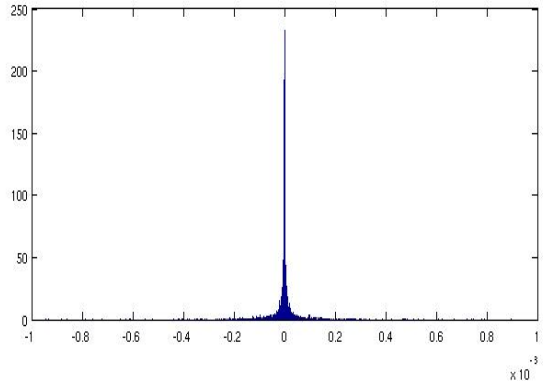


Figure 7: Min-Square error pdf for a single pattern. sensor *P'*

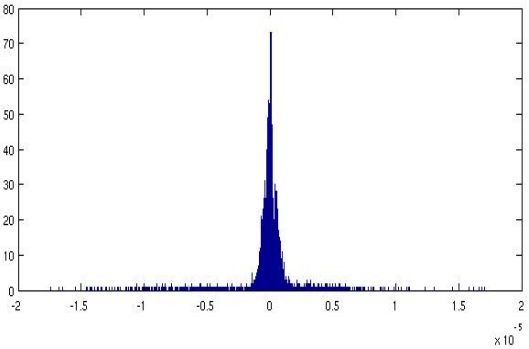


Figure 8: Max-Correntropy error pdf for a single pattern. sensor *R*

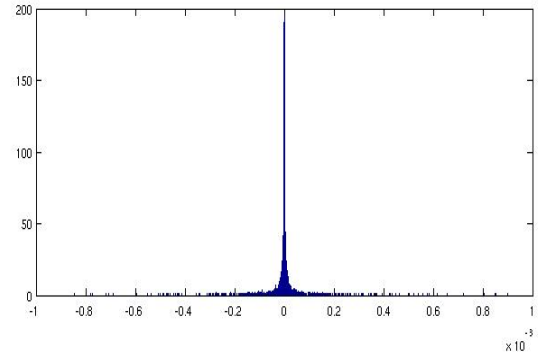


Figure 9: Min-Square error pdf for a single pattern. sensor *R*

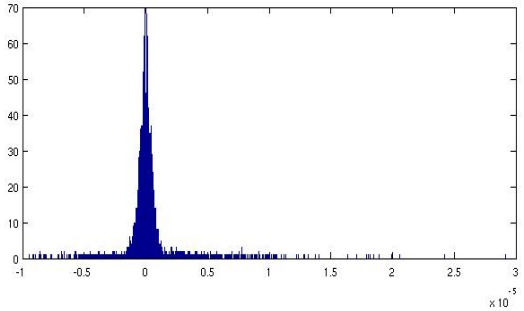


Figure 10: Max-Correntropy error pdf for a single pattern. sensor *P1*

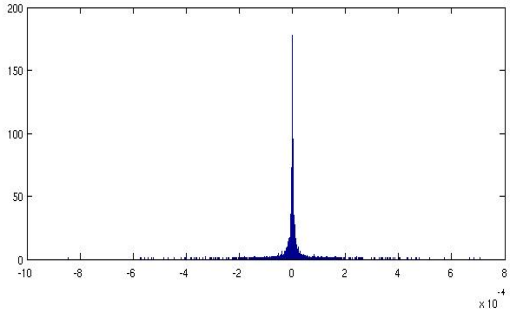


Figure 11: Min-Square error pdf for a single pattern. sensor *P1*

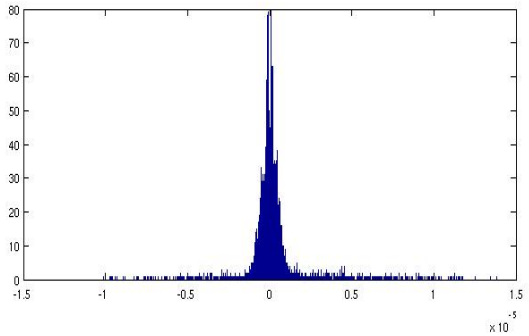


Figure 12: Max-Correntropy error pdf for a single pattern. sensor *P2*

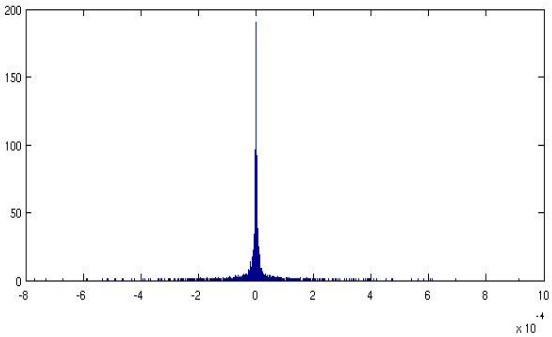


Figure 13: Min-Square error pdf for a single pattern. sensor *P2*

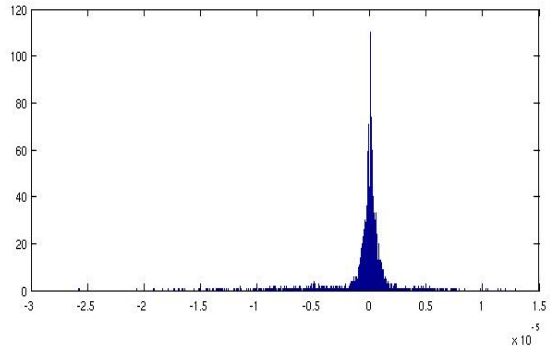


Figure 14: Max-Correntropy error pdf for a single pattern. sensor *P3*

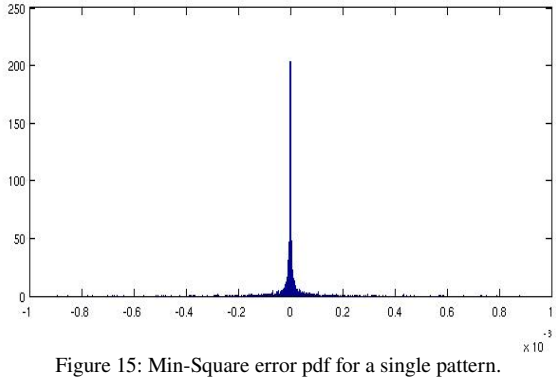


Figure 15: Min-Square error pdf for a single pattern. sensor *P3*

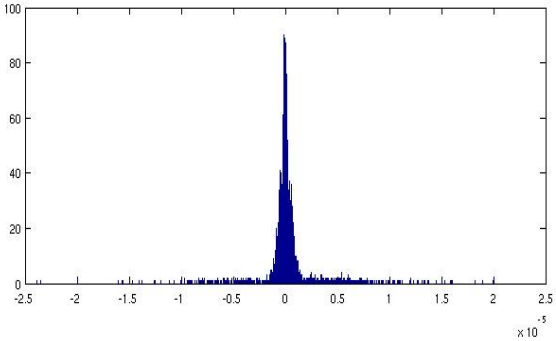


Figure 16: Max-Correntropy error pdf for a single pattern. sensor *P4*

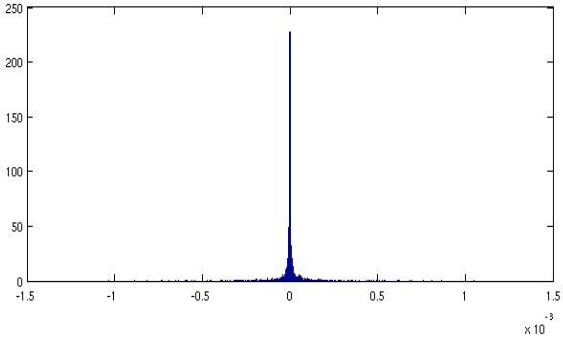


Figure 17: Min-Square error pdf for a single pattern. sensor *P4*

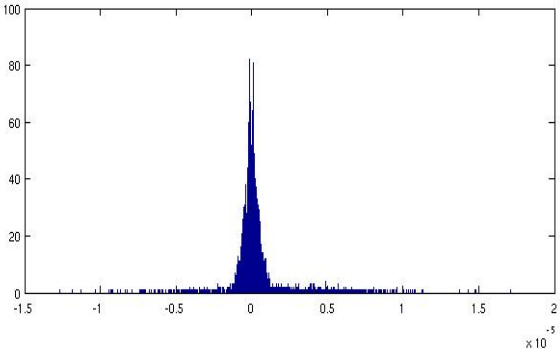


Figure 18: Max-Correntropy error pdf for a single pattern. sensor *P5*

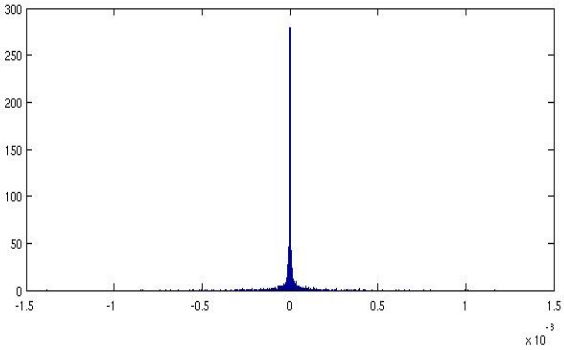


Figure 19: Min-Square error pdf for a single pattern. sensor *P5*

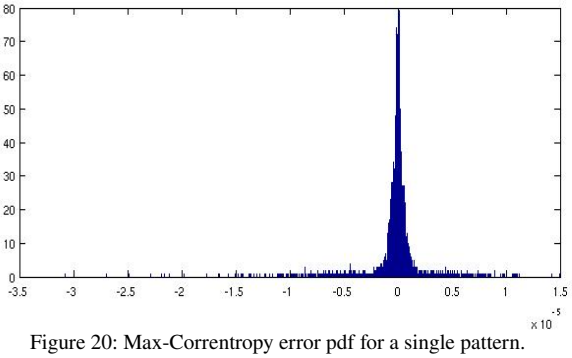


Figure 20: Max-Correntropy error pdf for a single pattern. sensor *P6*

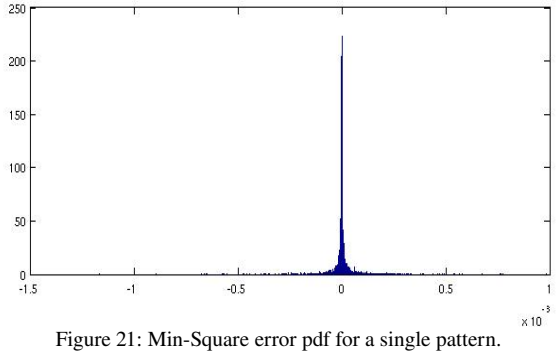


Figure 21: Min-Square error pdf for a single pattern. sensor *P6*

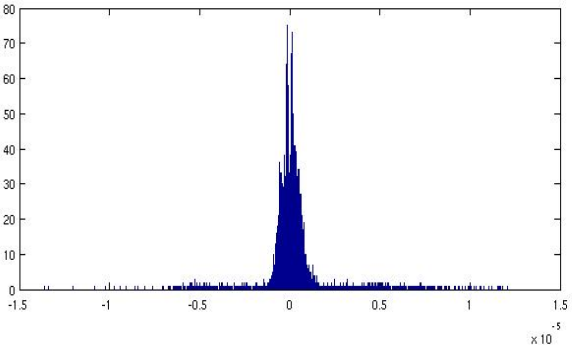


Figure 22: Max-Correntropy error pdf for a single pattern. sensor *P7*

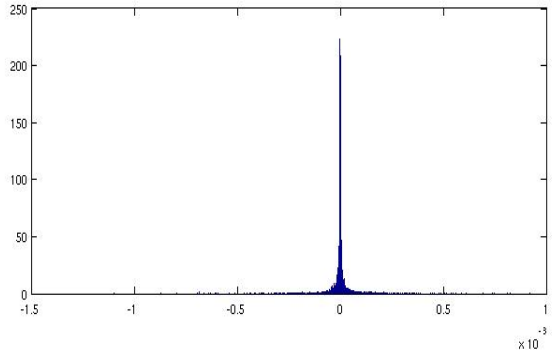


Figure 23: Min-Square error pdf for a single pattern. sensor *P7*

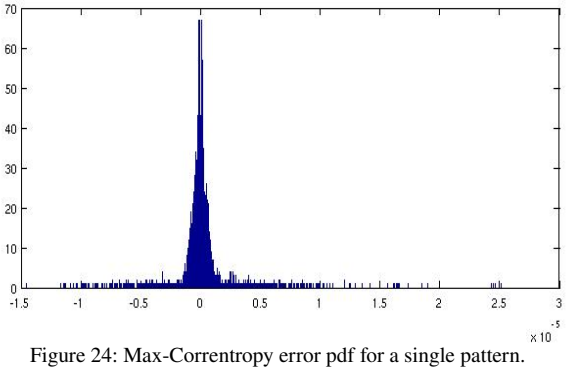


Figure 24: Max-Correntropy error pdf for a single pattern. sensor *P8*

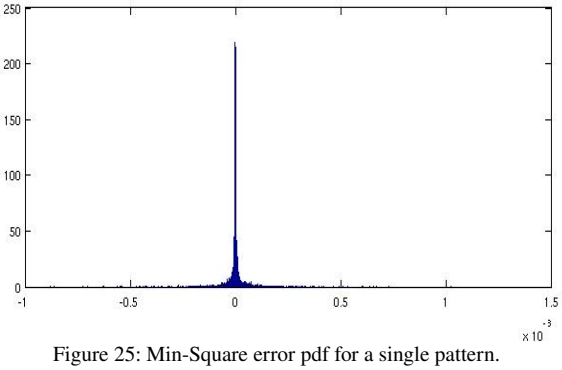


Figure 25: Min-Square error pdf for a single pattern. sensor *P8*

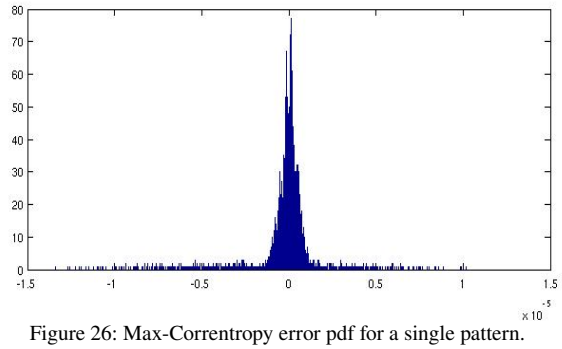


Figure 26: Max-Correntropy error pdf for a single pattern. sensor *P9*

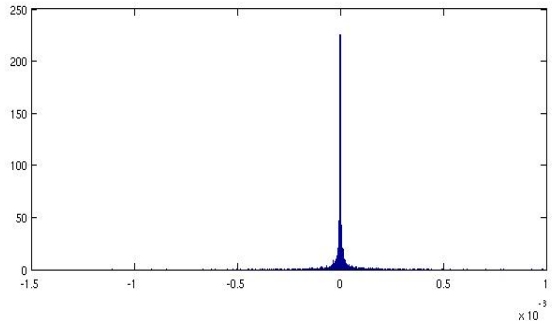


Figure 27: Min-Square error pdf for a single pattern. sensor *P9*

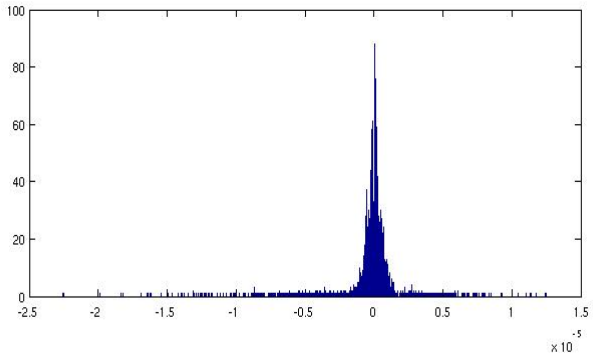


Figure 28: Max-Correntropy error pdf for a single pattern. sensor *P10*

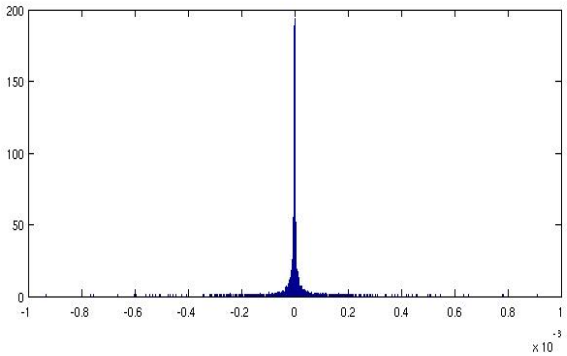


Figure 29: Min-Square error pdf for a single pattern. sensor *P10*

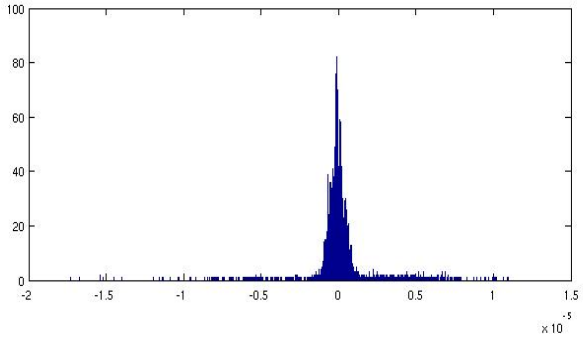


Figure 30: Max-Correntropy error pdf for a single pattern. sensor *P11*

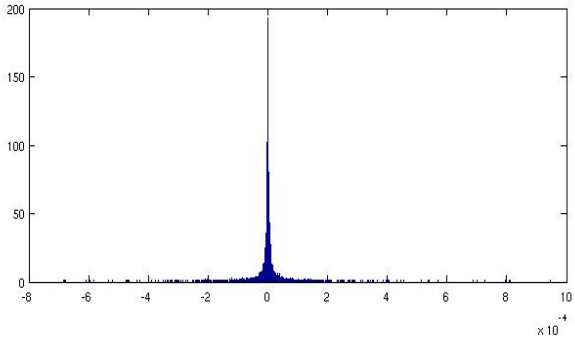


Figure 31: Min-Square error pdf for a single pattern. sensor *P11*

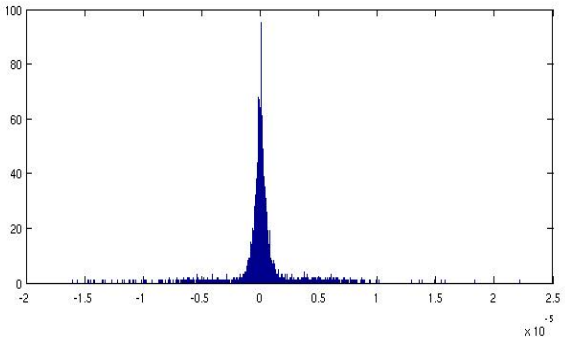


Figure 32: Max-Correntropy error pdf for a single pattern. sensor *P12*

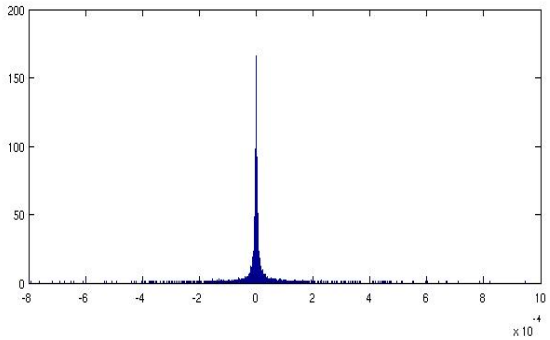


Figure 33: Min-Square error pdf for a single pattern. sensor *P12*

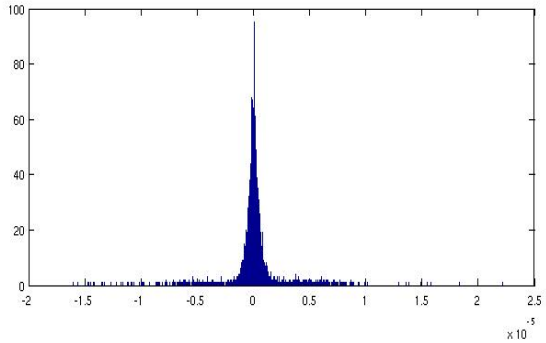


Figure 34: Max-Correntropy error pdf for a single pattern. sensor *S*

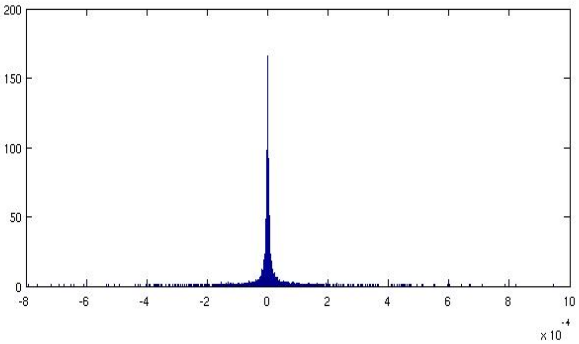


Figure 35: Min-Square error pdf for a single pattern. sensor *S*

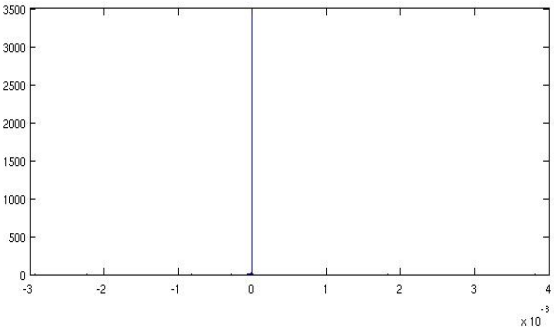


Figure 36: Max-Correntropy error pdf for a single pattern. sensor *D*

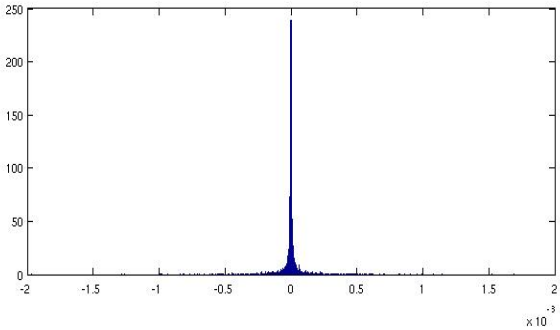


Figure 37: Min-Square error pdf for a single pattern. sensor *D*

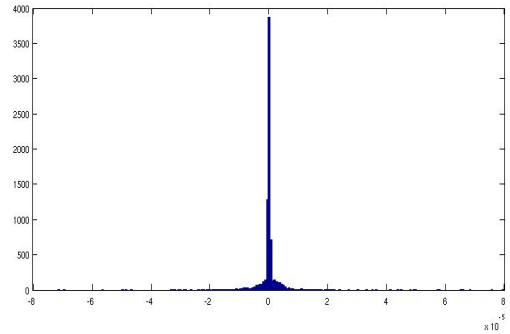

Figure 38: Max-Correntropy error pdf for concatenated patterns. sensor *P'*

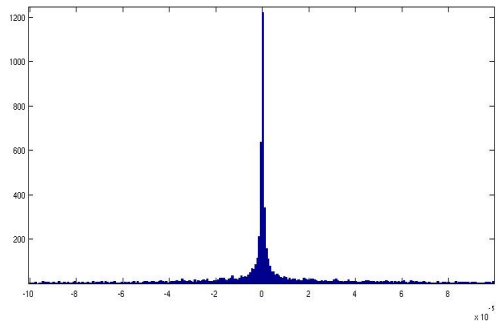

Figure 39: Min-Square error pdf for concatenated patterns. sensor *P'*

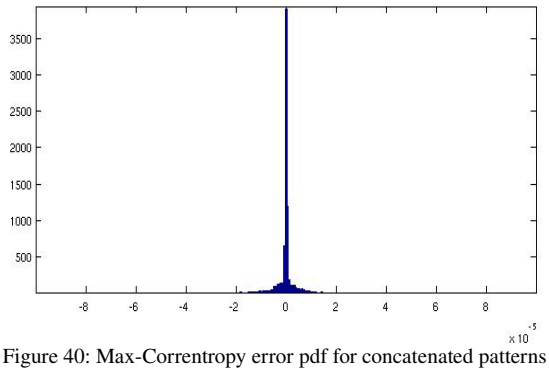

Figure 40: Max-Correntropy error pdf for concatenated patterns sensor *R*

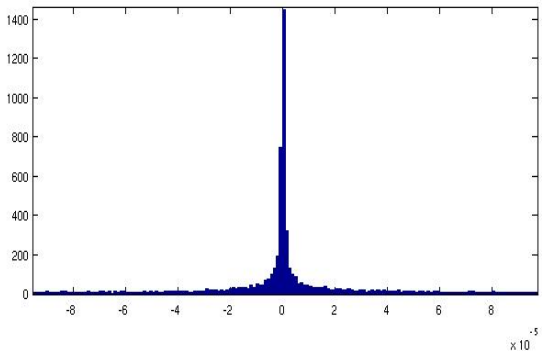

Figure 41: Min-Square error pdf for concatenated patterns. sensor *R*

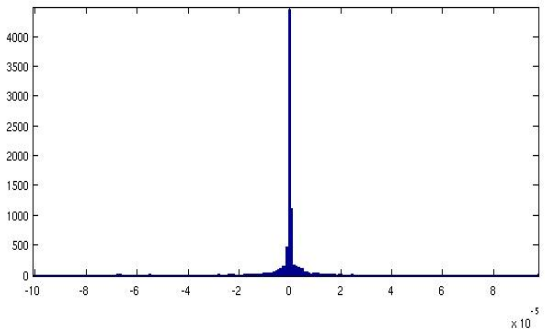

Figure 42: Max-Correntropy error pdf for concatenated patterns. sensor *P1*

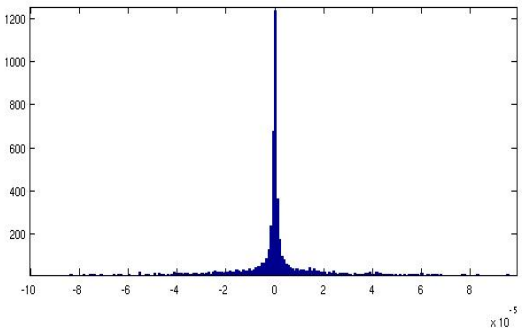

Figure 43: Min-Square error pdf for concatenated patterns. sensor *P1*

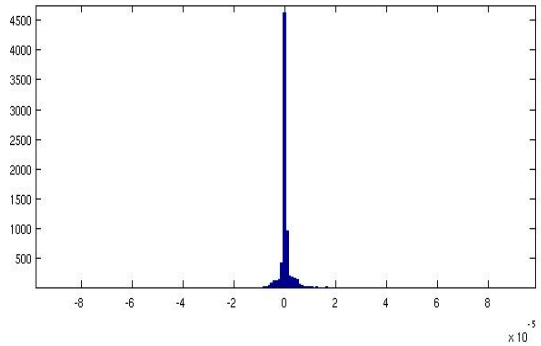

Figure 44: Max-Correntropy error pdf for concatenated patterns . sensor *P2*

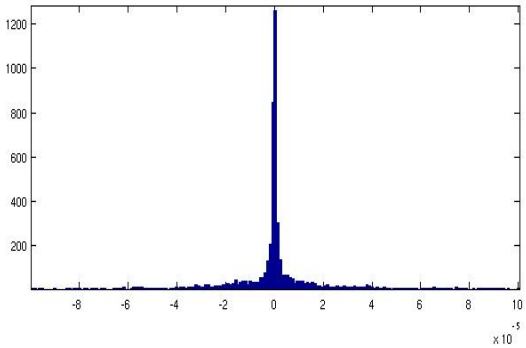

Figure 45: Min-Square error pdf for concatenated patterns. sensor *P2*

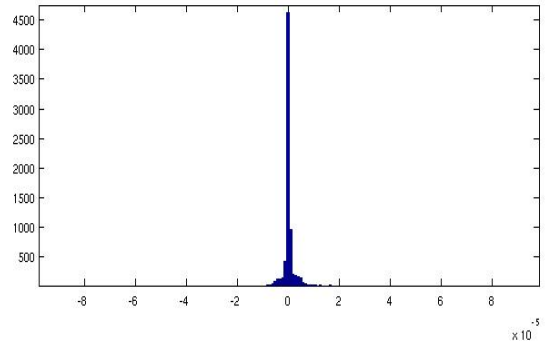


Figure 46: Max-Correntropy error pdf for concatenated patterns. sensor *P3*

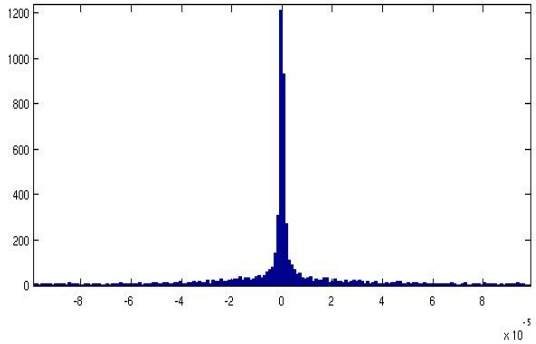


Figure 47: Min-Square error pdf for concatenated patterns. sensor *P3*

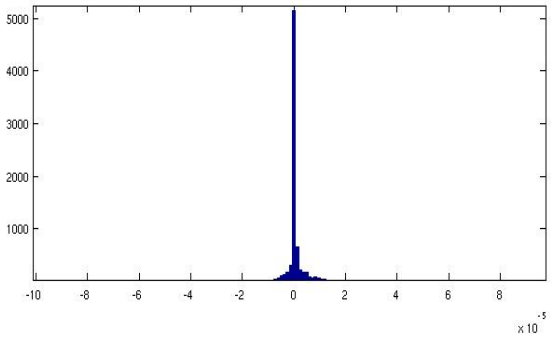


Figure 48: Max-Correntropy error pdf for concatenated patterns. sensor *P4*

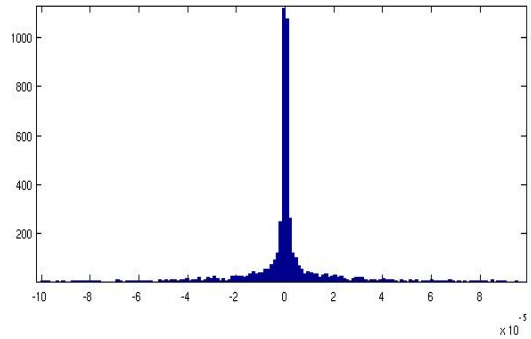


Figure 49: Min-Square error pdf for concatenated patterns. sensor *P4*

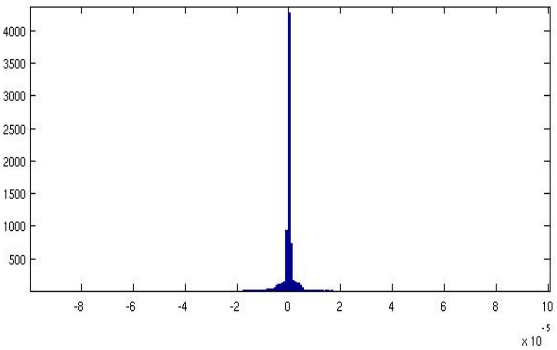


Figure 50: Max-Correntropy error pdf for concatenated patterns. sensor *P5*

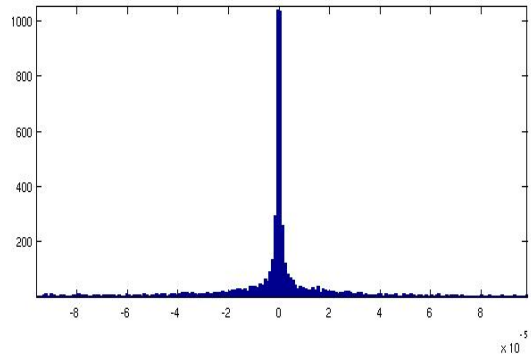


Figure 51: Min-Square error pdf for concatenated patterns. sensor *P5*

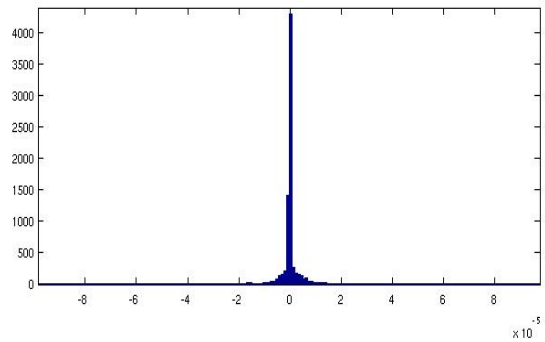


Figure 52: Max-Correntropy error pdf for concatenated patterns. sensor *P6*

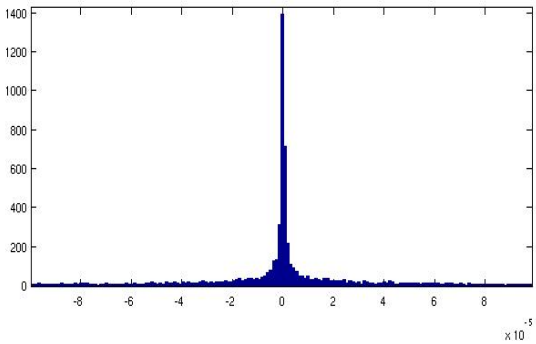


Figure 53: Min-Square error pdf for concatenated patterns. sensor *P6*

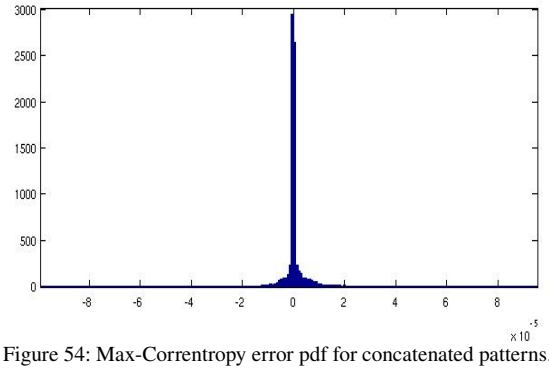


Figure 54: Max-Correntropy error pdf for concatenated patterns. sensor *P7*

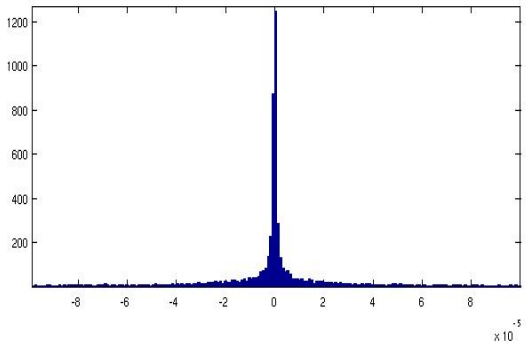


Figure 55: Min-Square error pdf for concatenated patterns. sensor *P7*

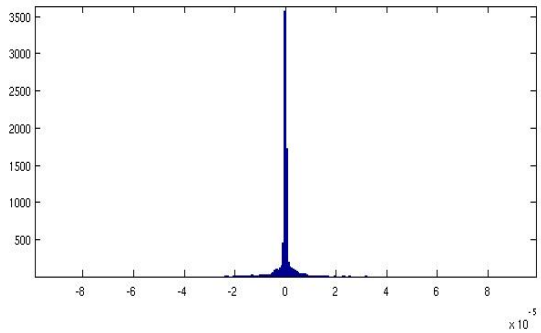


Figure 56: Max-Correntropy error pdf for concatenated patterns. sensor *P8*

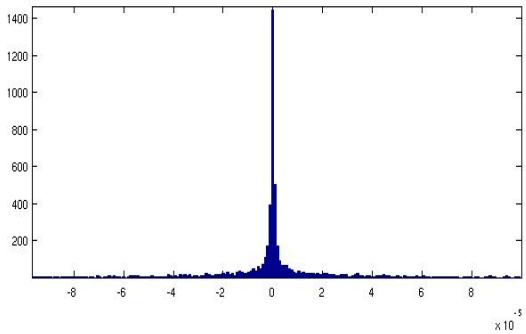


Figure 57: Min-Square error pdf for concatenated patterns. sensor *P8*

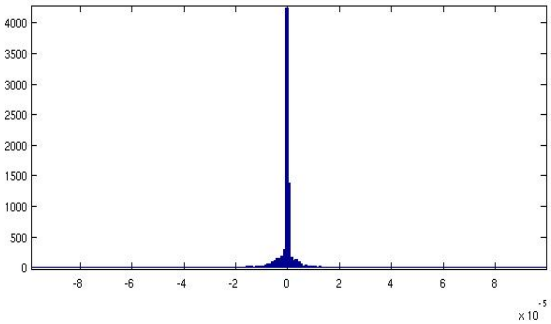


Figure 58: Max-Correntropy error pdf for concatenated patterns. sensor *P9*

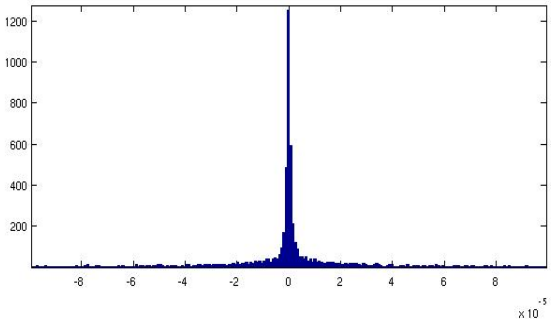


Figure 59: Min-Square error pdf for concatenated patterns. sensor *P9*

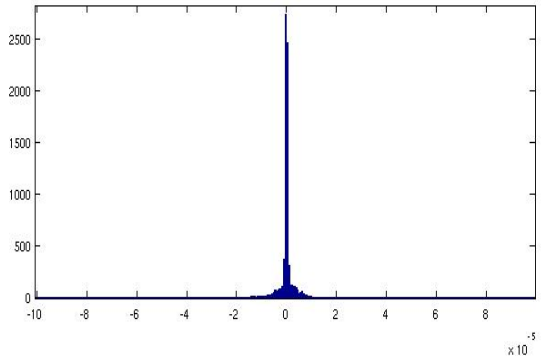


Figure 60: Max-Correntropy error pdf for concatenated patterns. sensor *P10*

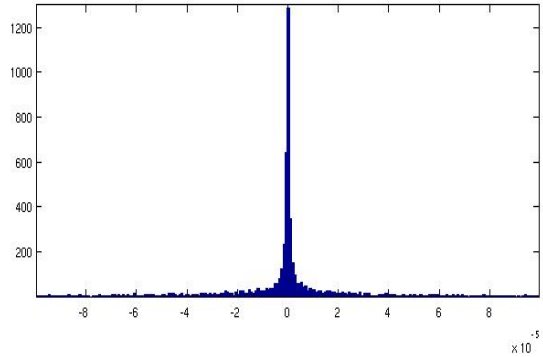


Figure 61: Min-Square error pdf for concatenated patterns. sensor *P10*

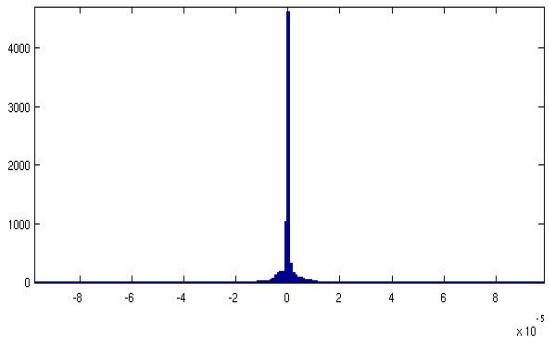


Figure 62: Max-Correntropy error pdf for concatenated patterns. sensor *P11*

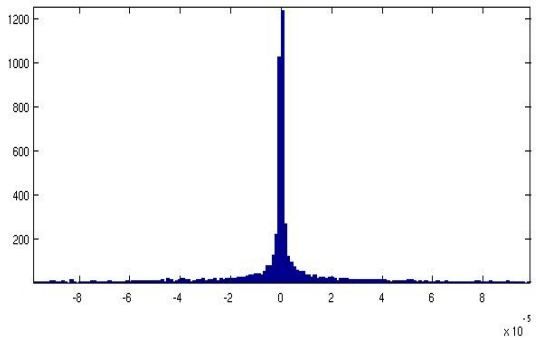


Figure 63: Min-Square error pdf for concatenated patterns. sensor *P11*

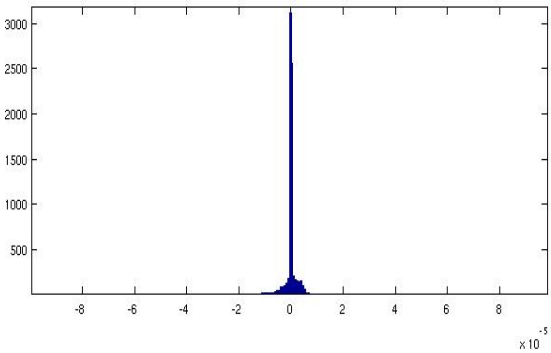


Figure 64: Max-Correntropy error pdf for concatenated patterns. sensor *P12*

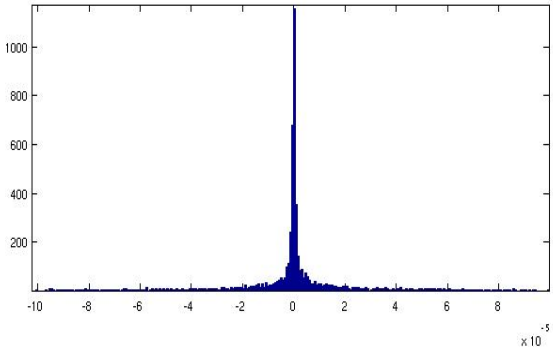


Figure 65: Min-Square error pdf for concatenated patterns. sensor *P12*

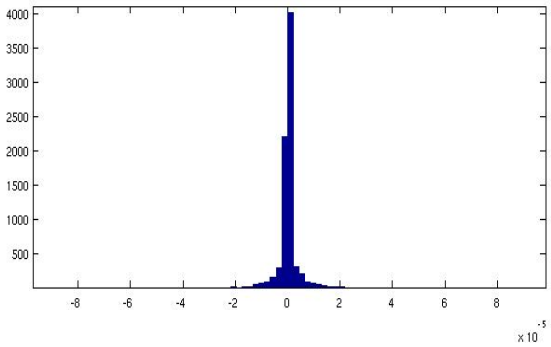


Figure 66: Max-Correntropy error pdf for concatenated patterns. sensor *S*

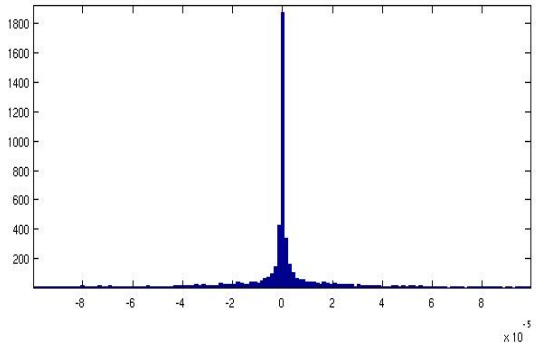


Figure 67: Min-Square error pdf for concatenated patterns. sensor *S*

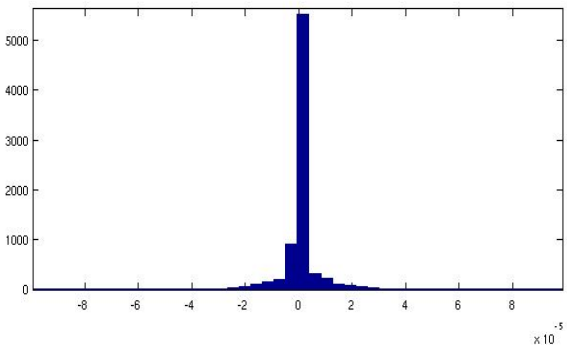


Figure 68: Max-Correntropy error pdf for concatenated patterns. sensor *D*

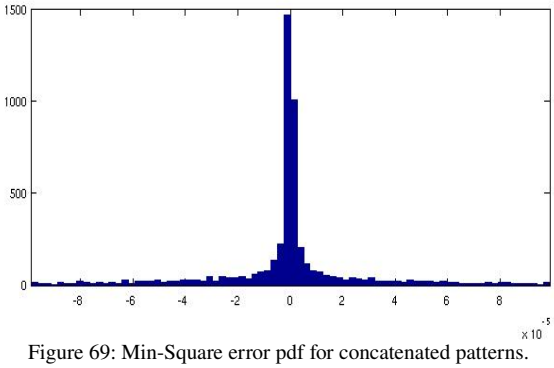


Figure 69: Min-Square error pdf for concatenated patterns. sensor *D*

## 4.6 Conclusion and discussion

This is a case study where one can state without doubt that “results speak for themselves”.

Correlation was found to exist between apparent chaotic eolic power central readings time series, satisfying the main objective of this particular study.

Let us now compare supervised learning methods.

By observation of error density functions, it's quite obvious the superiority of the error correntropy maximization criterion versus the mean square error minimization one.

For all sensor readings, the kernel's dispersion is always smaller for correntropy criterion trained auto-encoders, as one can conclude by consulting the following table:

| | Single pattern Max-Correntropy trained encoders | | Single pattern Min-Square error trained encoders | | Concatenated pattern Max-Correntropy trained encoders | | Concatenated pattern Min-Square error trained encoders | |
|---|---|---|---|---|---|---|---|---|
| Sensor | min(ε) | max(ε) | min(ε) | max(ε) | min(ε) | max(ε) | min(ε) | max(ε) |
| P' | -1.3649e$^{-05}$ | 9,6932e$^{-06}$ | -9.4480e$^{-04}$ | 8.9732e$^{-04}$ | -3.8823e$^{-03}$ | 2.2709e$^{-03}$ | -3.5621e$^{-03}$ | 4.1243e$^{-03}$ |
| R | -1.7405e$^{-05}$ | 1.7032e$^{-05}$ | -8.4518e$^{-04}$ | 8.9869e$^{-04}$ | -2.6780e$^{-03}$ | 3.8116e$^{-03}$ | -5.0246e$^{-03}$ | 6.2587e$^{-03}$ |
| $P_1$ | -9.4622e$^{-05}$ | 2.9132e$^{-05}$ | -8.4418e$^{-04}$ | 7.0661e$^{-04}$ | -4.8032e$^{-03}$ | 3.6344e$^{-03}$ | -4.7290e$^{-03}$ | 4.0753e$^{-03}$ |
| $P_2$ | -1.0118e$^{-05}$ | 1.3847e$^{-05}$ | -7.6929e$^{-04}$ | 9.1451e$^{-04}$ | -4.9989e$^{-03}$ | 4.7462e$^{-03}$ | -4.7240e$^{-03}$ | 5.4393e$^{-03}$ |
| $P_3$ | -2.5779e$^{-05}$ | 1,2882e$^{-05}$ | -8.9365e$^{-04}$ | 8.7854e$^{-04}$ | -4.5134e$^{-03}$ | 7.1169e$^{-03}$ | -4.8104e$^{-03}$ | 5.9996e$^{-03}$ |
| $P_4$ | -2.3887e$^{-05}$ | 2.0072e$^{-05}$ | -1.0320e$^{-03}$ | 1.0481e$^{-03}$ | -4.1750e$^{-03}$ | 7.7723e$^{-03}$ | -5.7690e$^{-03}$ | 6.6295e$^{-03}$ |
| $P_5$ | -1.2666e$^{-05}$ | 1.7077e$^{-05}$ | -1.3832e$^{-03}$ | 1.1637e$^{-03}$ | -3.7211e$^{-03}$ | 3.6754e$^{-03}$ | -5.2301e$^{-03}$ | 4.7385e$^{-03}$ |
| $P_6$ | -3.0763e$^{-05}$ | 1.4916e$^{-05}$ | -1.1662e$^{-03}$ | 9.8428e$^{-04}$ | -4.1976e$^{-03}$ | 6.2307e$^{-03}$ | -4.2952e$^{-03}$ | 6.8996e$^{-03}$ |
| $P_7$ | -1.3590e$^{-05}$ | 1.2107e$^{-05}$ | -1.0999e$^{-03}$ | 9.2237e$^{-04}$ | -5.1398e$^{-03}$ | 3.3138e$^{-03}$ | -4.6978e$^{-03}$ | 5.0627e$^{-03}$ |
| $P_8$ | -1.4558e$^{-05}$ | 2.5108e$^{-05}$ | -8.7884e$^{-04}$ | 1.0260e$^{-03}$ | -3.0684e$^{-03}$ | 3.4534e$^{-03}$ | -5.3002e$^{-03}$ | 4.2041e$^{-03}$ |
| $P_9$ | -1.3366e$^{-05}$ | 1.0163e$^{-05}$ | -1.1063e$^{-03}$ | 9.8195e$^{-04}$ | -5.1811e$^{-03}$ | 3.8962e$^{-03}$ | -4.5486e$^{-03}$ | 3.9620e$^{-03}$ |
| $P_{10}$ | -2.2531e$^{-05}$ | 1.2457e$^{-05}$ | -9.3179e$^{-04}$ | 9.1080e$^{-04}$ | -2.1569e$^{-03}$ | 3.7767e$^{-03}$ | -4.2323e$^{-03}$ | 4.3835e$^{-03}$ |
| $P_{11}$ | -1.7285e$^{-05}$ | 1.0930e$^{-05}$ | -9.2002e$^{-04}$ | 9.3823e$^{-04}$ | -5.6895e$^{-03}$ | 3.9434e$^{-03}$ | -3.9115e$^{-03}$ | 7.6985e$^{-03}$ |
| $P_{12}$ | -1.1712e$^{-05}$ | 1.0185e$^{-05}$ | -7.9934e$^{-04}$ | 9.4664e$^{-04}$ | -4.2553e$^{-03}$ | 2.8738e$^{-03}$ | -3.8502e$^{-03}$ | 3.5452e$^{-03}$ |
| S | -1.6004e$^{-05}$ | 2.2173e$^{-05}$ | -7.9890e$^{-04}$ | 9.4602e$^{-04}$ | -8.9905e$^{-03}$ | 1.2668e$^{-02}$ | -6.2490e$^{-03}$ | 7.5779e$^{-03}$ |
| D | -2.9271e$^{-05}$ | 3.8140e$^{-03}$ | -1.9644e$^{-03}$ | 1.6906e$^{-03}$ | -2.2368e$^{-02}$ | 2.1081e$^{-02}$ | -1.2650e$^{-02}$ | 1.2072e$^{-02}$ |

**Table 1: Limits for the absolute error for error correntropy maximization and mean square error minimization based learnings.**

Exemplifying histograms state quite clearly error correntropy maximization criterion effectiveness, consistently tending to maximize error probability density around the origin, while mean square error minimization tends to spread error density distribution around the origin, but in a more widely manner.

Although both training resulting error histograms show a pronounced spike around the origin, correntropy maximization criterion is able to place more than the double of the error information particles around the origin, sometimes in a 100 times smaller sized interval, defined by:

$$L_i = \left[ -\left( \frac{max(\varepsilon_i) - min(\varepsilon_i)}{1000} \right), \left( \frac{max(\varepsilon_i) - min(\varepsilon_i)}{1000} \right) \right]$$

Where $\varepsilon_i$ is the concatenated $i^{th}$ sensor error vector.

More surprisingly, when comparing mean square error for both cases, error correntropy maximization surpasses mean square minimization method's training ability.

| Sensor | Single pattern Max-Correntropy trained encoders | Single pattern Min-Square error trained encoders | Concatenated pattern Max-Correntropy trained encoders | Concatenated pattern Min-Square error trained encoders |
|---|---|---|---|---|
| P' | 1.6234e$^{-12}$ | 3.2754e$^{-09}$ | 1.3674e$^{-08}$ | 1.9208e$^{-07}$ |
| R | 3.1407e$^{-12}$ | 3.2002e$^{-09}$ | 1.7245e$^{-08}$ | 2.5282e$^{-07}$ |
| $P_1$ | 2.7430e$^{-12}$ | 2.4122e$^{-09}$ | 1.7428e$^{-08}$ | 2.3321e$^{-07}$ |
| $P_2$ | 2.3107e$^{-12}$ | 2.6109e$^{-09}$ | 3.7435e$^{-08}$ | 2.9541e$^{-07}$ |
| $P_3$ | 3.7852e$^{-12}$ | 3.0894e$^{-09}$ | 4.5832e$^{-08}$ | 2.7443e$^{-07}$ |
| $P_4$ | 3.2082e$^{-12}$ | 3.5721e$^{-09}$ | 5.8305e$^{-08}$ | 3.8841e$^{-07}$ |
| $P_5$ | 2.1701e$^{-12}$ | 3.4668e$^{-09}$ | 2.9475e$^{-08}$ | 2.5828e$^{-07}$ |
| $P_6$ | 3.5722e$^{-12}$ | 3.1581e$^{-09}$ | 4.2462e$^{-08}$ | 2.9447e$^{-07}$ |
| $P_7$ | 2.3631e$^{-12}$ | 3.6060e$^{-09}$ | 4.0927e$^{-08}$ | 2.7828e$^{-07}$ |
| $P_8$ | 3.2048e$^{-12}$ | 2.9915e$^{-09}$ | 2.3286e$^{-08}$ | 2.4141e$^{-07}$ |
| $P_9$ | 2.1936e$^{-12}$ | 3.1117e$^{-09}$ | 2.4646e$^{-08}$ | 2.4375e$^{-07}$ |
| $P_{10}$ | 3.0249e$^{-12}$ | 2.8637e$^{-09}$ | 1.3016e$^{-08}$ | 2.3007e$^{-07}$ |
| $P_{11}$ | 2.0816e$^{-12}$ | 3.0835e$^{-09}$ | 3.4931e$^{-08}$ | 2.7352e$^{-07}$ |
| $P_{12}$ | 1.5193e$^{-12}$ | 2.7972e$^{-09}$ | 1.3237e$^{-08}$ | 1.8070e$^{-07}$ |
| S | 2.7443e$^{-12}$ | 3.6194e$^{-09}$ | 1.5030e$^{-07}$ | 6.1631e$^{-07}$ |
| D | 4.0340e$^{-09}$ | 8.2737e$^{-09}$ | 1.0042e$^{-06}$ | 1.5130e$^{-06}$ |

**Table 2 : Max-correntropy and mean square error trained auto-encoders per-sensor output mean square error.**

Let us not forget that these errors were calculated for normalized inputs. Still, normalized inputs resolution are:

| P' | R | $P_1$ | $P_1$ | $P_1$ | $P_1$ | $P_1$ | $P_1$ | $P_1$ | $P_1$ | $P_1$ | $P_1$ | $P_1$ | $P_1$ | S | D |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $10^{-5}$ | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | $10^{-4}$ | $10^{-3}$ | $10^{-3}$ |

**Table 3 : Max-correntropy and mean square error auto-encoders per-sensor normalized resolution.**

It is clearly perceivable that none of the selected algorithms were able to guarantee all output errors below resolution's threshold for the number of iterations imposed, although, some individual max-correntropy criterion trained auto-encoders fulfilled such condition. This shows some pattern information dependency. This means that for successful max-correntropy cases, a 6 days journey was successfully recovered without introducing more error than the one associated with sensor resolution.

It became quite clear, during training sessions, that by using correntropy maximization one could improve these results with few more iterations. The same wasn't verified for the contestant mean square error minimization method, that achieved the presented results with considerable effort.

One can state that training auto-encoders by means of error correntropy maximization surpassed all expectations, showing a more robust behaviour by rendering globally superior

solutions than min-square error algorithm with the same ease of use. In fact, once found a particular set of successive training gains and kernel sizes, this set can be used to satisfyingly train similar auto-encoders, not demanding at the same time similar data.

Max-correntropy auto-encoding is also more efficient, demonstrating such efficiency by attaining better results with less training iterations.

Lastly, successful data encoding implies data compression in the hidden layer and hidden to output connection synapses.

More significant is the reduction of the auto-encoder's input size to *1/10* of the original one, if one uses the hidden layer as an input one, thus attaining data compression goals for both algorithms.

In conclusion, one must recognize max-correntropy criterion undoubtful superiority.

# Chapter 5

## Application of min-entropy auto-encoders to missing signal restoration

## 5.1 Introduction

Throughout this chapter one will discuss signal restoration techniques and introduce a new restoration method, starting by explaining a simple application of alternating projections into convex sets (POCS).

Before proceeding with such exposition, one must remember some basic definitions to demonstrate how can auto-encoders help us to recover lost or corrupt signals.

Previous works (1) state that for a fully trained auto-encoder there is an unique convergence point for a given operating point, composed by a set of known inputs.

As said in (1), this implies that the trained auto-encoders implements contractive mapping, which is a sufficient condition for the convergence of an iterative approach to signal restoration. Moreover, it is also affirmed that such convergence point should be reached independently of missing signal initialization values.

Let us now review the definition of contraction.

A contractive mapping, defined according to (12) and (13), is a mapping $M : X \to X$ on a complete metric space *(X,d)* in which, for any pair *(x,y)* in that space:

$$d(M_x, M_y) \leqslant k\, d(x, y)$$
$$0 \leqslant k < 1$$

Therefore, a contraction mapping ensures that the distance between two outputs is always less than the distance between inputs.

In an Euclidean norm metric space in $\mathbb{R}^1$ , *M* is a functional mapping $g(x)$ where:

$$\|g(x) - g(y)\| \leqslant k \|x - y\|$$

Making *y* equal to *x*+*dx* yields:

$$\frac{\|g(x) - g(x+dx)\|}{\|dx\|} \leqslant k$$

The limit of this expression as $dx \to 0$ is:

$$\left| \frac{dg(x)}{dx} \right| < 1$$

This is in fact a less strict requisite for contraction, meaning that it would be sufficient to demonstrate for some function *g(x)* that the derivative is less than unity for all *x*. Application of the *Banach Fixed-Point Theorem* under such contraction conditions yields a unique fixed point $x_0$ for which $g(x_0) = x_0$ guaranteeing convergence for any element $x_{n+1} = g(x_n)$ contemplated in sequence $\{x_n\}$ (1).

However, a multiple pattern perfectly trained auto-encoder ensures that the *Banach Fixed-Point Theorem* is not applicable.

Enlarging the contractive mapping space limits to:

$$0 \leqslant k \leqslant 1$$

The mapper is now a non-expansive one. Auto-encoders work as a non-expansive convex orthogonal projection mapper and the operation $x_{n+1}=g(x_n)$ will still converge to a fixed point, although this fixed point will now depend on the selected initial point (1).

Initial point solution dependency and corresponding non-uniqueness problem has to be taken in consideration to understand neither strictly contractive nor non-expansive auto-encoder behaviour.

Although, one can demonstrate similarity between trained auto-encoder behaviour, contractiveness and non-expansiveness properties.

Despite auto-encoder cannot be considered fully contractive nor non-expansive as a whole, at some operating point it may be contractive once it operates on a subset of the input vector $I_m$ (1).

As said before, to ensure non-expansiveness the derivative for all sensor values must be less than one. Setting the interval [0,1] for the input samples by some kind of input normalization is quite straightforward, and once the discrete derivative is:

$$\Delta g_x(x) = g(x) - g(x-1)$$

It is evident that respective derivatives will satisfy contractiveness condition, once the perfectly trained auto-encoder mapper $g(x)$ enforces its outputs to be the same as its inputs.

Concluding, a perfectly trained auto-encoder under the conditions herein described demonstrates contractive properties by permitting more than one missing sensor restoration, independently of the chosen operating point.

## 5.2 Signal restoration using POCS

Narayanan et al. (15) describes projection onto convex sets *(POCS)* , an evident solution for missing signal restoration by means of a trained auto-encoder, assumed to be convex.

In fact, if one assumes convexity for all input pattern sets, meaning, known sensors $x_k$, and a second convex set containing randomly distributed known and missing input sensors $x_m$, forward propagation of this missing sensors vector will generate the auto-encoder output vector $\hat{x}$ which is a projection $W_1$ onto the first set.

Once convexity is ensured, swaping the input vector $x_m$ with the output vector $\hat{x}$ assures convergence to a solution which is an intersection of the two sets, the projection $W_2$.

Alternating successively $W_1$ and $W_2$ completes a POCS algorithm iteration.

## 5.3 Signal restoration using error minimization algorithms

In (1), unconstrained search is also suggested to contour lack of convexity issues, if one is motivated to find better solutions than POCS.

The described unconstrained search can be based on any error minimization method to discover the true point of convergence for the auto-encoder outputs corresponding to those missing sensor values:

$$\min_{x_m} \| x_m - \hat{x_m} \|$$

But such algorithm would imply previous knowledge of which are the missing sensor signals to be restored.

Real world applications may not know which are the corrupt signals to be recovered. Just picture the situation where a sensor is doing incorrect readings, though its readings by themselves do not denounce any kind of malfunction. In this case, the auto-encoder should reveal discrepancy between input and respective output signals, but it might not be so, specially for a non-perfectly trained auto-encoder. Hence, instead of using only the information relative to missing sensor using an unconstrained search, one can do a constrained search using the complete output error to improve the final answer (1):

$$\min_{x_m} \| x - \hat{x} \|$$

This allows a smoother match between input and output and helps eliminating incongruous solutions that don't make sense on the known sensor context.

Hence, mean square error minimization criterion can be used to perform the intended unconstrained search. The adjustment to be done after each iteration will be:

$$\Delta\, x_m = x_m - \hat{x_m} = \varepsilon_m$$

The next iteration input will be:

$$x'_m = \hat{x_m} + \gamma\, \Delta x_m$$

Where $\gamma$ is an adjustable gain. This calculation must be done for each output and input pair separately.

Even though, constrained search needs to have sufficient information to accurately reconstruct lost signals.

## 5.4 Signal restoration using error correntropy maximization

If one can utilize any error minimization rule to perform the constrained search (1), inherent error correntropy maximization method tendency to minimize the output error, by maximizing the error probability density function around the origin, suggests this rule ability to restore missing signals. In this case, the new criterion for the constrained search is:

$$\max_{x_m} \| J(x-\hat{x}) \|$$

Defining $J(x-\hat{x})$ as the max-correntropy for the difference between input and output, estimated using Parzen windowing:

$$J(x-\hat{x}) = \frac{1}{M} \sum_{m=1}^{M} k_\sigma \left( g(x,w) - x_m \right)$$

To maximize this expression, one can apply a gradient descent technique, yielding for each sensor the following adjustment value:

$$\Delta' x_m = -\left( \frac{-dJ(x_m)}{d x_m} \right) = \left( \frac{x_m - \hat{x_m}}{\sqrt{2\pi}\,\sigma^3} \right) e^{-\left( \frac{(x_m - \hat{x_m})^2}{2\sigma^2} \right)}$$

The next iteration input will then be:

$$x'_m = \hat{x_m} + \gamma\, \Delta' x_m$$

## 5.5 Problem Presentation

Recovering missing or corrupt sensor signals sent by some remotely located eolic central to a control central occured over a given period of time would allow postponing to a more convenient moment in time any required action to amend sensor malfunctioning.

Signal restoration objectives will be pursued using both min-entropy and min-square error trained auto-encoders which set the different initial operating points for both constrained search algorithms. Both constrained search methods will be applied to both trained type auto-encoders to permit performance comparison and, hopefully, to unveil some interesting combination of efforts between mentioned supervised learning algorithms.

Resuming, to compare constrained search missing signal restoration performances based on mean square error minimization and error correntropy maximization is the main objective of the work detailed in the remainder of this chapter.

## 5.6 Problem formulation

Problem formulation can be resumed to restore successively failing random sensors, which output signals do not evidently reveal any sort of dysfunction. This must be done over an approximate 30 consecutive days period.

To simulate sensor malfunction, arbitraryly chosen values substituted correct sensor readings. The randomly selected failing sensors were $P'$, $R$, $P_3$, $P_5$, and $P_6$, nearly *1/3* of the total input size for a given time instant $t$.

Although a larger number of failing sensors was tested, the obtained results contained more spurious values than correctly restored data, rendering globally meaningless answers. This was expected to happen, proving the final solution's quality dependency of the number of known sensors. Hence, these results were excluded from this analysis.

A total of 5 input patterns containing successively increasing misreading sensors were tested. Each pattern covers a 6 days time period. The complete group of patterns covers the desired 30 days period. Once sensor failing is to be sequential and cumulative, patterns were arranged to satisfy these demands. Thus, pattern number one, has one failing sensor, namely $P'$, pattern number two, has two failing sensors, namely $P'$ and $R$, and so on, until last pattern is reached, which has the complete set of malfunctioning sensors, $P'$, $R$, $P_3$, $P_5$, and $P_6$.

The structure adopted to organize the input vectors is the same as the consubstantiated in chapter 4.4, thus facilitating reuse of already trained auto-encoders.

A set of 10 auto-encoders was employed to assist in signal recovery, half of which trained to maximize output's error correntropy, the other half trained to minimize the output's square error.

Each auto-encoder was trained to map one of the known sensor patterns, as explained in chapter 4.5. Obviously, auto-encoders will try to auto-associate the correspondent mixed known and unknown sensors input.

For example, consider auto-encoder $E_1$ which encodes pattern $A_1$, composed of 25 input vectors $I_m$, each one singly composed of known inputs:

$$I_m \equiv \{P'_1, R_1, P_{11}, \ldots, P_{121}, S_1, D_1, \ldots, P'_n, R_n, \ldots, P_{12n}, S_n, D_n\}^T$$

Let us now define correspondent faulty pattern $A'_1$, composed by input vectors $I'_m$, which integrate some corrupted data.

Supposing that the sensor responsible for registering the overall produced power $P'$ is the one flickering:

$$I'_m \equiv \{\phi(P'_1), R_1, P_{11}, \ldots, P_{121}, S_1, D_1, \ldots, \phi(P'_n) + p_n, R_n, \ldots, P_{12n}, S_n, D_n\}^T$$

Where $\phi(P'_n)$ may well be a function which emulates the faulty sensor $P'_n$ behaviour. For this case study, arbitrary values were selected within the normalized space defined in chapter 4.5 to simulate bad readings $\phi(P'_n)$. Sensor signal restoration will be done for both algorithms using $A'_1$ as initial operating point and auto-encoder $E_1$ to generate the correspondent outputs by forward propagation. These outputs will then be used for the first missing signal restoration training iteration, according to the theoretic approach exposed in preceding chapters 5.3 and 5.4.

## 5.7 Simulations

One can devise two major training sets depending on the chosen point of view.

Output error probability distribution functions were drawn for both pointed missing signal restoration supervised learning methods, using either max-correntropy, either min-square error trained auto-encoders. This first group of simulations allows us to discuss the advantages of using one or the other, by evaluating which performs best independently of the mappers weights and biases.

On the other hand, knowing how these weights were set, that is, by maximizing error correntropy or by minimizing the samples square error, one can also examine which type of mapper is better tuned to recover missing signals.

Training under information correntropy maximization criterion was realized with large Parzen windows due to the extremely small amount of training data, consubstantiated on a single sample per operating point. By large understand nearly 1000 times bigger than maximum error.

In order to maintain objective function output scaled to reasonable numeric values for both cases, the gain parameter was adjusted in conformance.

Let us now resume what did we get for the tests realized under the specified terms.

Note that only the global per-sensor results are herein shown. Consequently, the chosen sets used to print the diverse shown histograms concatenate the mentioned 5 input patterns error vector. These error vectors were calculated by comparing the restored signals vector $I_m''$ with correspondent target vector $I_m$.

Remind that the maximum size for the auto-encoders input and output used for simulations was set to be 512 neurons, equivalent to 32 consecutively read sets $S_t$, which corresponds to an approximate 6h period. The trained auto-encoders correctly encode a 6 consecutive days journey, approximately, this meaning that each one satisfies simultaneously 25 input sets $I_m$.

Stop criterion for both supervised learning methods was defined to be the maximum absolute error $|\varepsilon_i|$ between input $I_i$ and respective forward propagated output $O'_i$ bounded to a maximum defined by:

$$|\varepsilon_m| = |I_m - O'_m| \leq 10^{-3}$$

To improve convergence, two important measures were undertaken, namely stop criterion smoothing and on-the-fly signal restore.

By on-the-fly signal restore, let us understand immediate forwarding propagation of the new complete array of inputs, as soon as a change on a single input sensor is done.

On the other hand, stop criterion always started at 0.01 and whenever successive iterations rendered a solution which all inputs satisfied such criterion, stop flag was reduced by an amount of 0.001. This helped us attaining better results than the ones attained without taking in consideration these concerns.

To allow the evaluation of any improvement of the error probability density distribution functions pending training, let us start by plotting the initial patterns input errors histograms which are to be improved by both refered constrained research methods (figures 70 to 85). These per-sensor histograms were plot including the samples registered over the intended 30 days period.

Once again, check graphic scales to avoid misinterpretation.

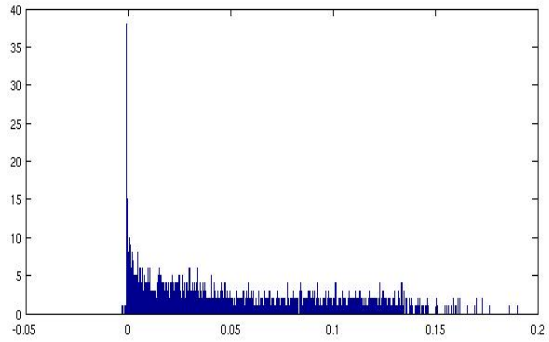


Figure 70: Initial training pattern pdf.
Missing sensor $P'$

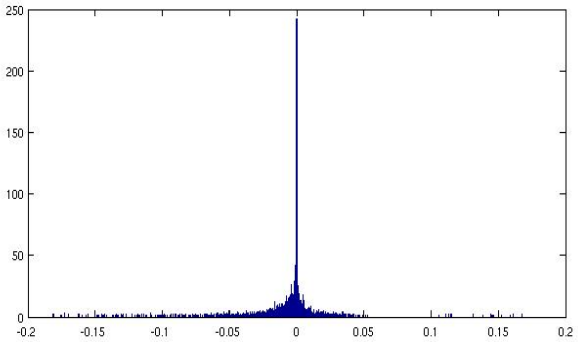


Figure 71: Initial training pattern pdf.
Missing sensor $R$

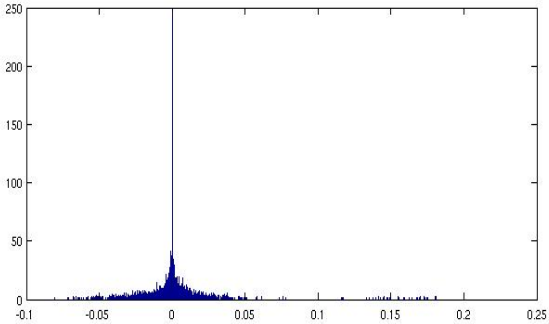


Figure 72: Initial training pattern pdf.
Known sensor $P_1$

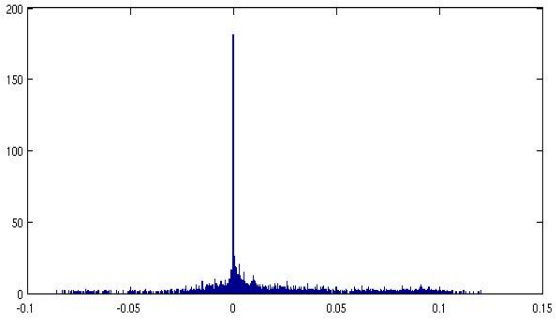


Figure 73: Initial training pattern pdf.
Known sensor $P_2$

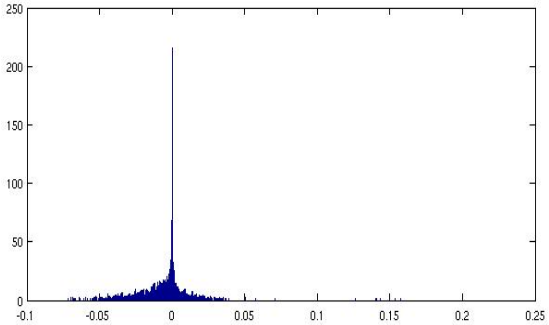


Figure 74: Initial training pattern pdf.
Missing sensor $P_3$

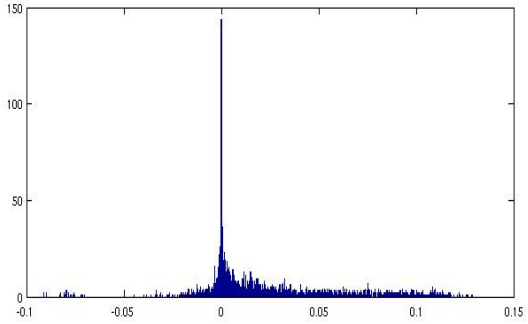


Figure 75: Initial training pattern pdf.
Known sensor $P_4$

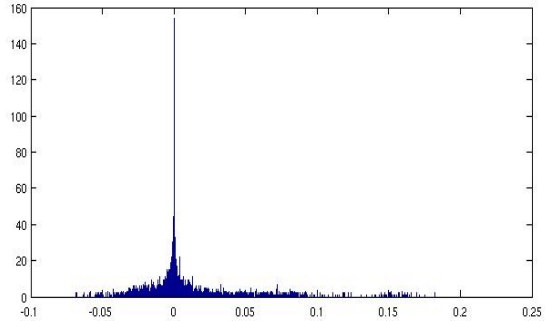


Figure 76: Initial training pattern pdf.
Missing sensor $P_5$

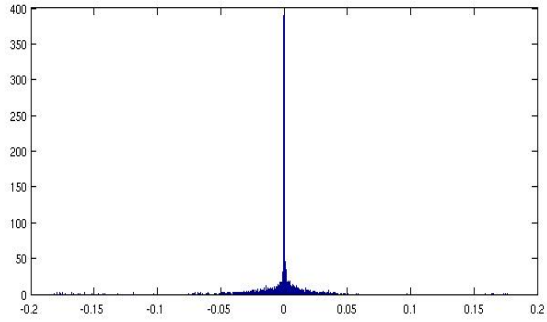


Figure 77: Initial training pattern pdf.
Missing sensor $P_6$

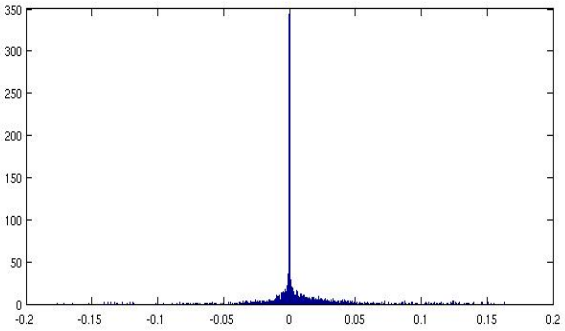


Figure 78: Initial training pattern pdf.
Known sensor $P_7$

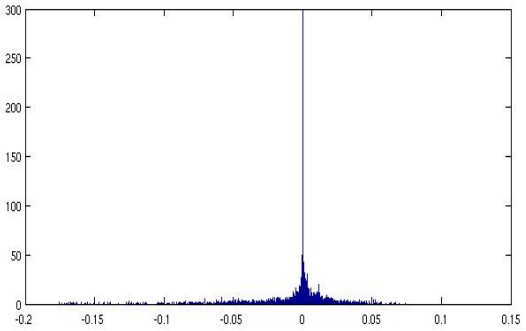


Figure 79: Initial training pattern pdf.
Known sensor $P_8$

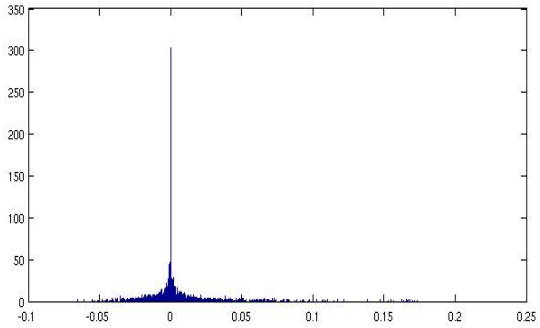


Figure 80: Initial training pattern pdf.
Known sensor $P_9$

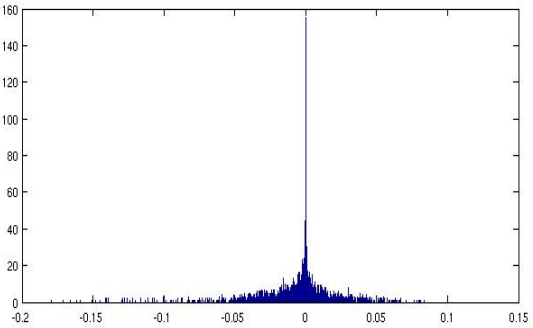


Figure 81: Initial training pattern pdf.
Known sensor $P_{10}$

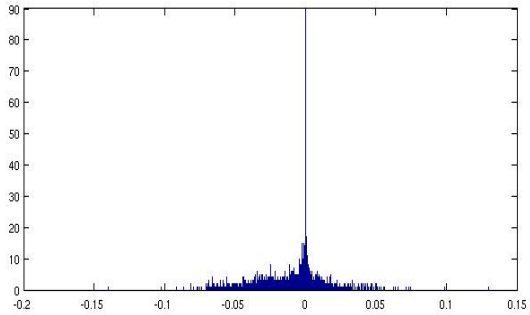


Figure 82: Initial training pattern pdf.
Known sensor $P_{11}$

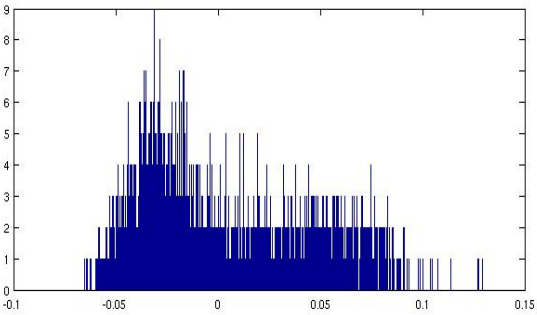


Figure 83: Initial training pattern pdf.
Known sensor $P_{12}$

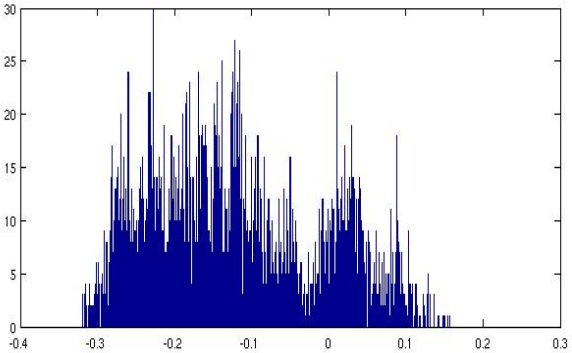


Figure 84: Initial training pattern pdf.
Known sensor $S$

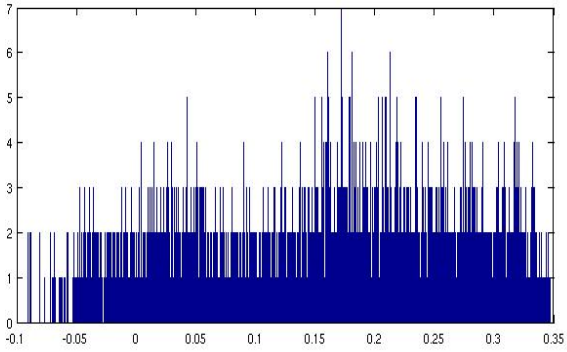


Figure 85: Initial training pattern pdf.
Known sensor $D$

Solution error probability density functions for the malfunctioning sensors $P'$, $R$, $P_3$, $P_5$, and $P_6$ are plotted in the following figures (86 to 105).

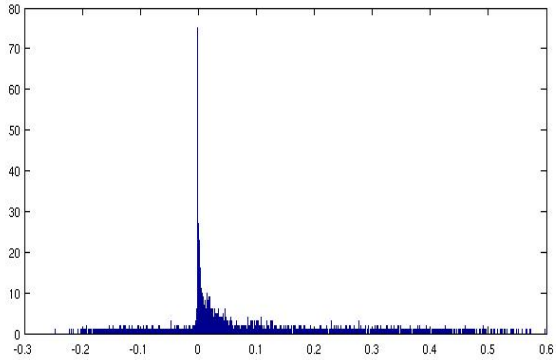


Figure 86: Max-Correntropy error pdf for Max-Correntropy trained auto-encoder.
Missing sensor $P'$

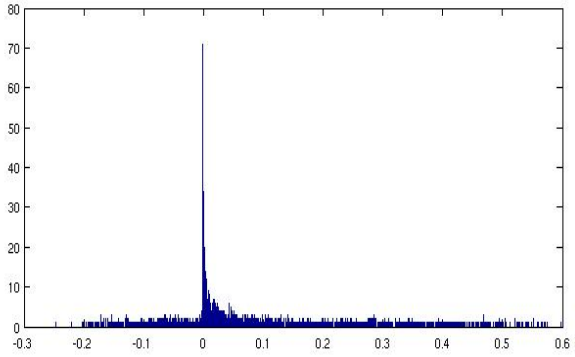


Figure 87: Min-Square error pdf for Max-Correntropy trained auto-encoder.
Missing sensor $P'$

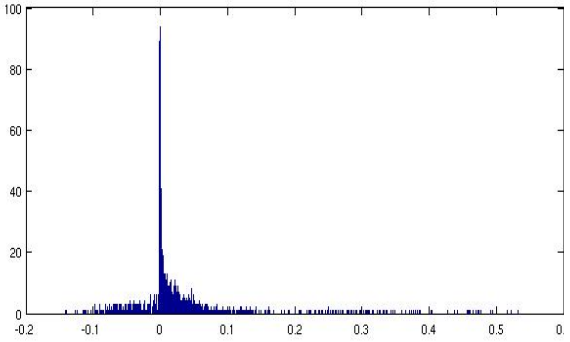


Figure 88: Max-Correntropy error pdf for Min-Square error trained auto-encoder.
Missing sensor $P'$

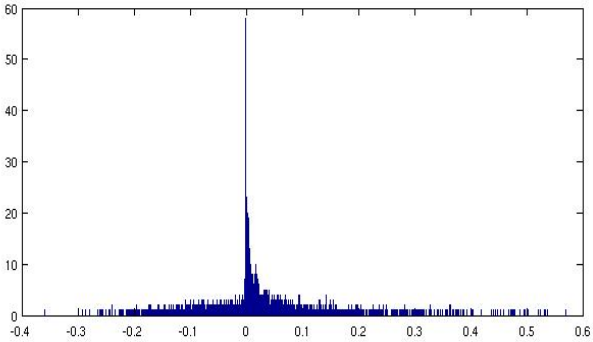


Figure 89: Min-Square error pdf for Min-Square error trained auto-encoder.
Missing sensor $P'$

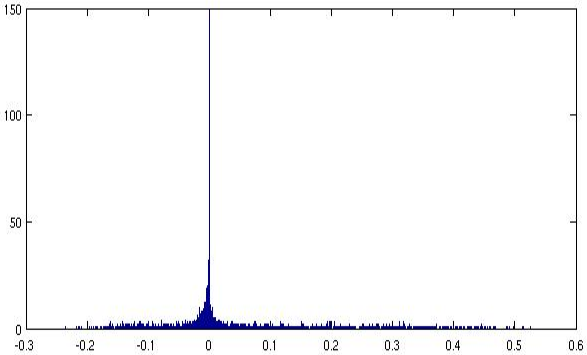


Figure 90: Max-Correntropy error pdf for Max-Correntropy trained auto-encoder.
Missing sensor $R$

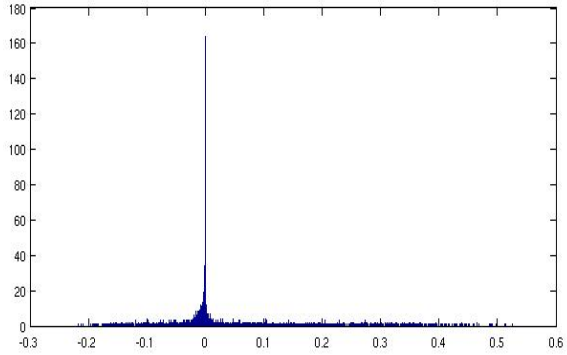


Figure 91: Min-Square error pdf for Max-Correntropy trained auto-encoder.
Missing sensor $R$

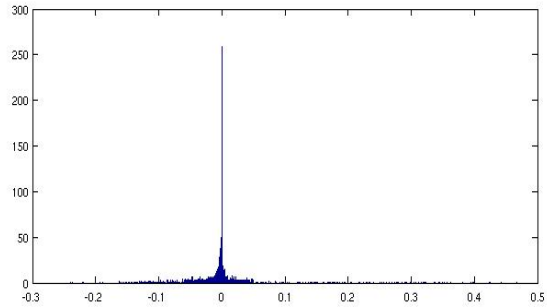


Figure 92: Max-Correntropy error pdf for Min-Square error trained auto-encoder.
Missing sensor *R*

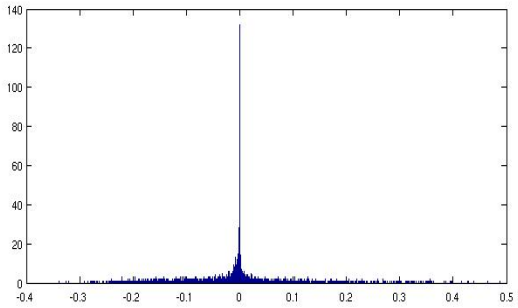


Figure 93: Min-Square error pdf for Min-Square error trained auto-encoder.
Missing sensor *R*

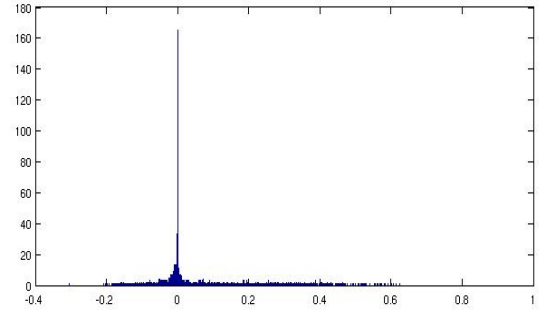


Figure 94: Max-Correntropy error pdf for Max-Correntropy trained auto-encoder.
Missing sensor $P_3$

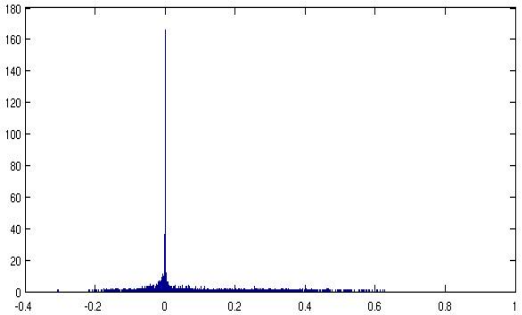


Figure 95: Min-Square error pdf for Max-Correntropy trained auto-encoder.
Missing sensor $P_3$

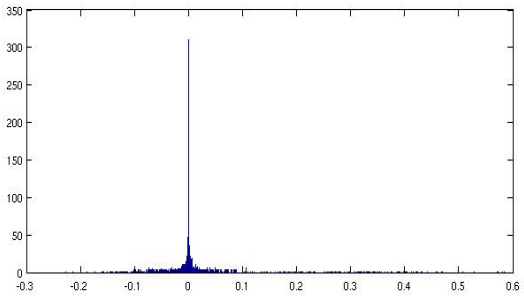


Figure 96: Max-Correntropy error pdf for Min-Square error trained auto-encoder.
Missing sensor $P_3$

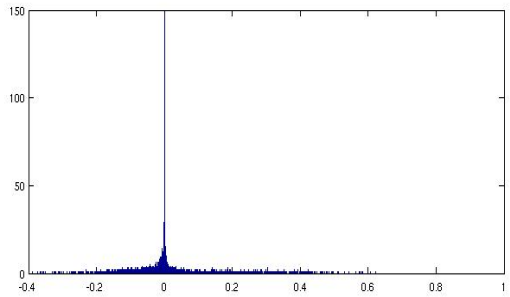


Figure 97: Min-Square error pdf for Min-Square error trained auto-encoder.
Missing sensor $P_3$

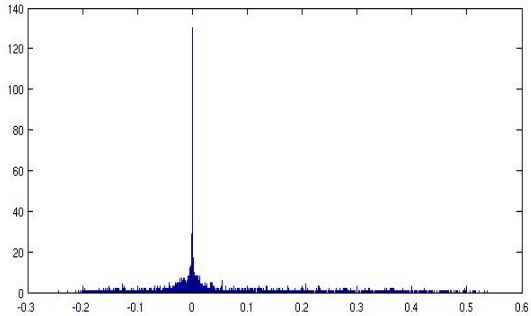


Figure 98: Max-Correntropy error pdf for Max-Correntropy trained auto-encoder.
Missing sensor $P_5$

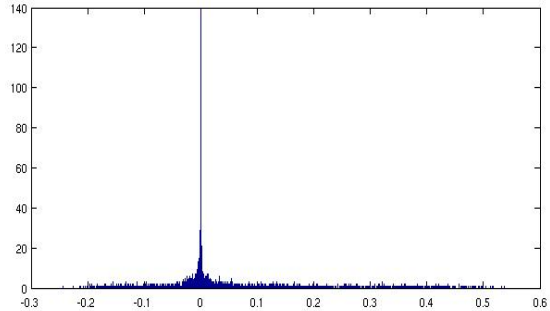


Figure 99: Min-Square error pdf for Max-Correntropy trained auto-encoder.
Missing sensor $P_5$

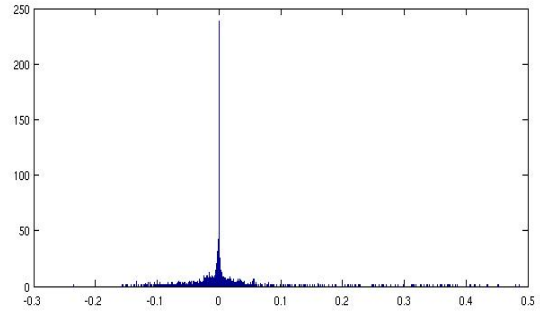


Figure 100: Max-Correntropy error pdf for Min-Square error trained auto-encoder.
Missing sensor $P_5$

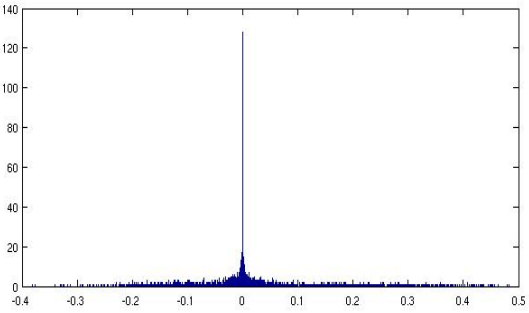


Figure 101: Min-Square error pdf for Min-Square error trained auto-encoder.
Missing sensor $P_5$

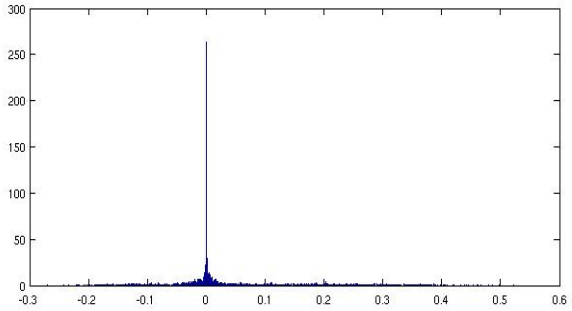


Figure 102: Max-Correntropy error pdf for Max-Correntropy trained auto-encoder.
Missing sensor $P_6$

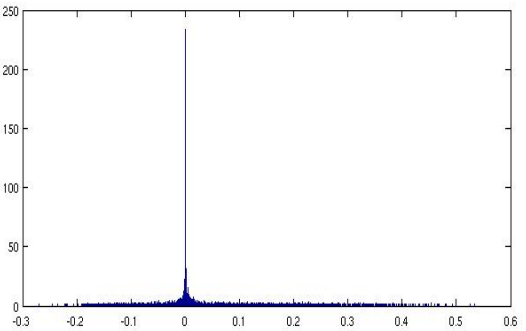


Figure 103: Min-Square error pdf for Max-Correntropy trained auto-encoder.
Missing sensor $P_6$

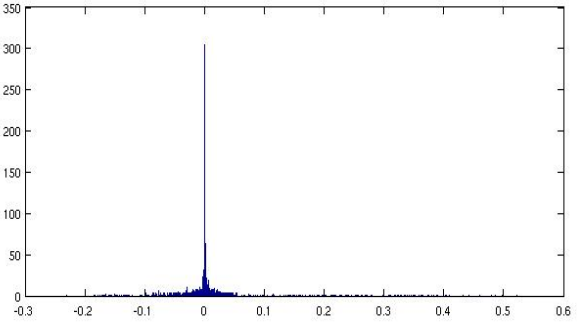


Figure 104: Max-Correntropy error pdf for Min-Square error trained auto-encoder.
Missing sensor $P_6$

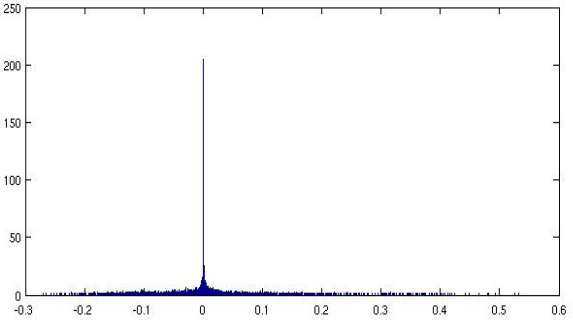


Figure 105: Min-Square error pdf for Min-Square error trained auto-encoder.
Missing sensor $P_6$

Focus now on the following figures 106 and 107 which represent part of one of the auto-encoders used. The neurons on the left (1 to 512) are the input neurons, the middle layer is the hidden layer (neurons 513 to 563) and on the right the output neurons (564 to 1076). Obviously, from an auto-encoder point of view, output 564 mirrors input 1, output 565 mirrors input 2 and so on.

Consider the effect that wrong signals have on known input encoded signals, which translates to increased encoded output error, clearly visible on the exemplifying picture 107, where only sensor *P'* family (neuron number 1 on the figure) is incorrect.

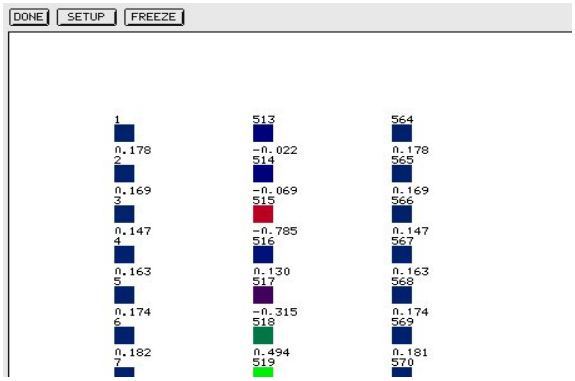


Figure 106: Targeted solution.

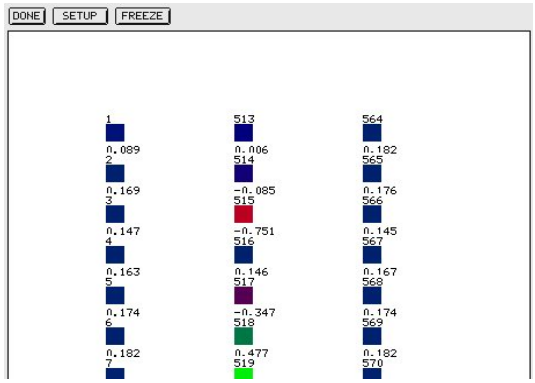


Figure 107: Missing sensor *P'* induced output error on known outputs.

Subsequently, the best and worst error probability density functions for known sensors were too herein included (figures 108 to 129), placing side by side the best and worst solution for each sensor.

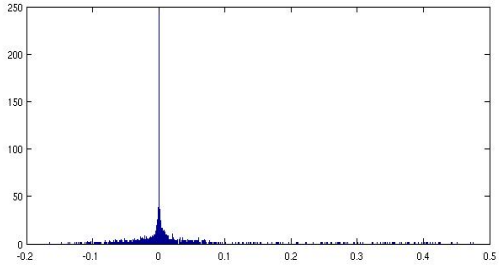


Figure 108: Best solution; Max-Correntropy error pdf for Min-Square error trained auto-encoder.
Known sensor $P_1$

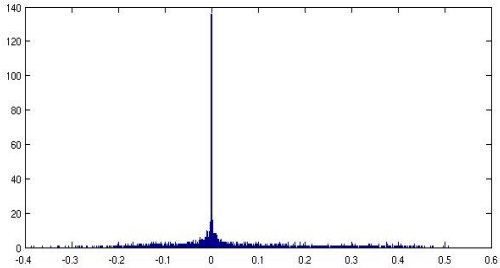


Figure 109: Worst Solution; Min-Square error pdf for Min-Square error trained auto-encoder.
Known sensor $P_1$

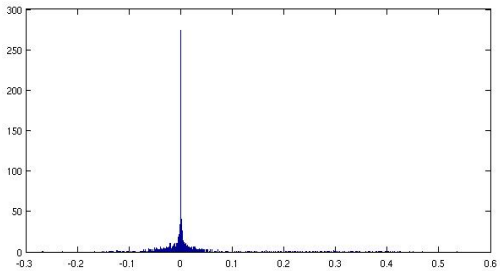


Figure 110: Best solution; Max-Correntropy error pdf for Mean-Square error trained auto-encoder.
Known sensor $P_2$

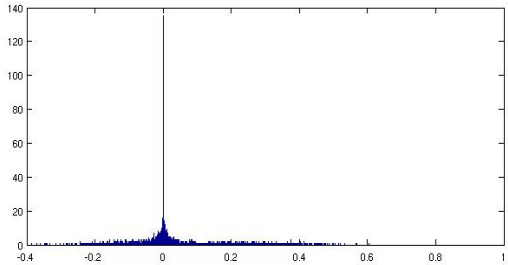


Figure 111: Worst Solution; Min-Square error pdf for Min-Square error trained auto-encoder.
Known sensor $P_2$

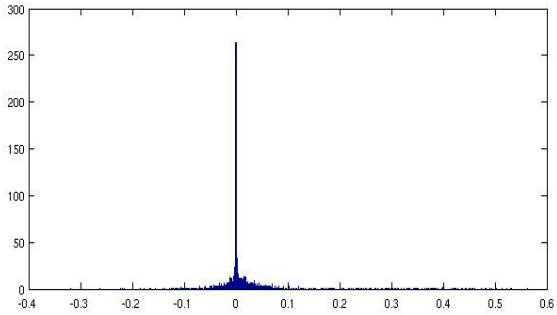


Figure 112: Best solution; Max-Correntropy error pdf for Mean-Square error trained auto-encoder.
Known sensor $P_4$

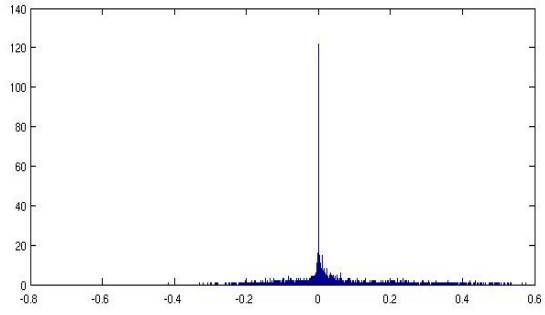


Figure 113: Worst Solution; Min-Square error pdf for Min-Square error trained auto-encoder.
Known sensor $P_4$

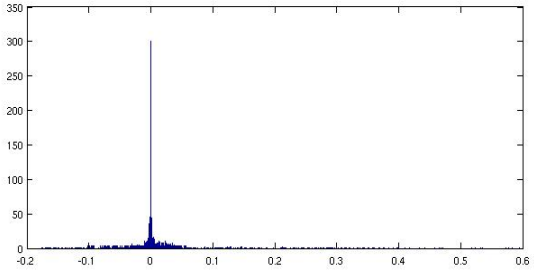


Figure 114: Best solution; Max-Correntropy error pdf for Mean-Square error trained auto-encoder.
Known sensor $P_7$

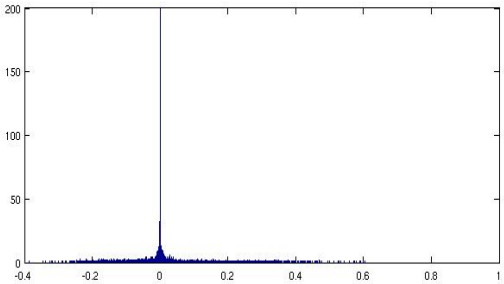


Figure 115: Worst Solution; Min-Square error pdf for Min-Square error trained auto-encoder.
Known sensor $P_7$

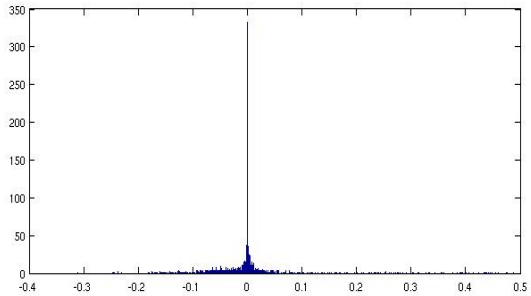


Figure 116: Best solution; Max-Correntropy error pdf for Mean-Square error trained auto-encoder.
Known sensor $P_8$

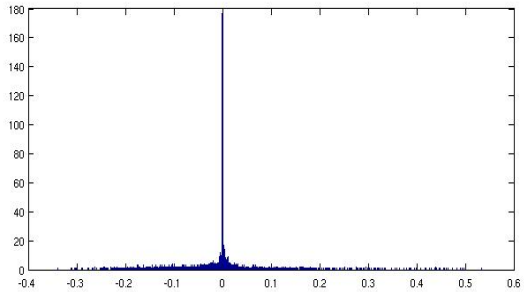


Figure 117: Worst Solution; Min-Square error pdf for Min-Square error trained auto-encoder.
Known sensor $P_8$

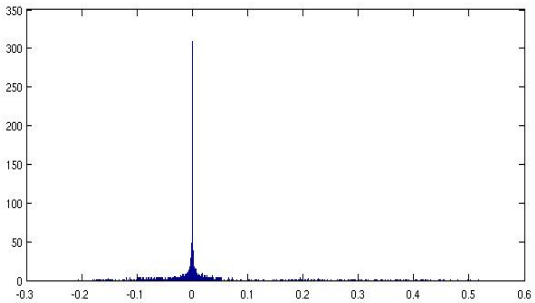


Figure 118: Best solution; Max-Correntropy error pdf for Mean-Square error trained auto-encoder.
Known sensor $P_9$

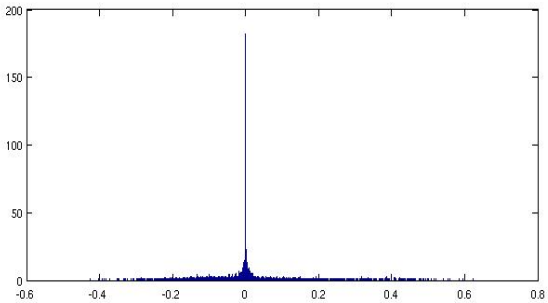


Figure 119: Worst Solution; Min-Square error pdf for Min-Square error trained auto-encoder.
Known sensor $P_9$

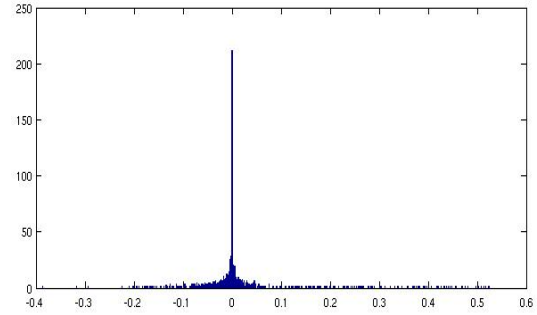


Figure 120: Best solution; Max-Correntropy error pdf for Mean-Square error trained auto-encoder.
Known sensor $P_{I0}$

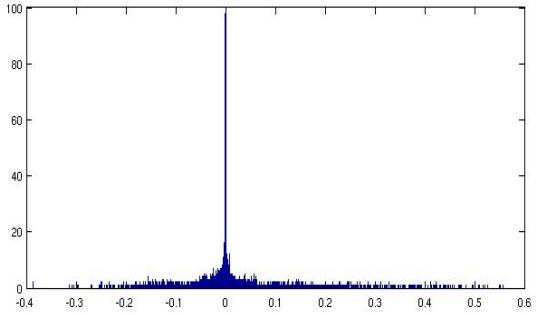


Figure 121: Worst Solution; Min-Square error pdf for Min-Square error trained auto-encoder.
Known sensor $P_{I0}$

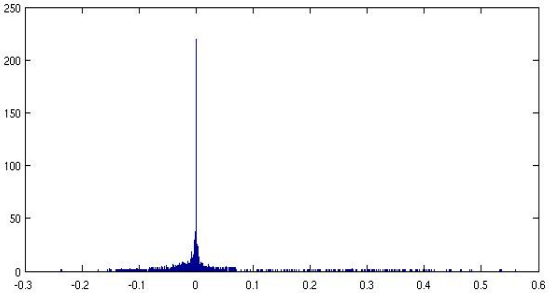


Figure 122: Best solution; Max-Correntropy error pdf for Mean-Square error trained auto-encoder.
Known sensor $P_{I1}$

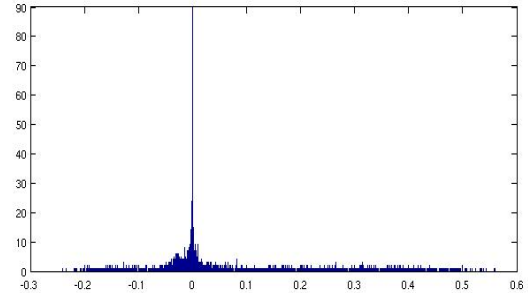


Figure 123: Worst Solution; Max-Correntropy error pdf for Max-Correntropy error trained auto-encoder.
Known sensor $P_{I1}$

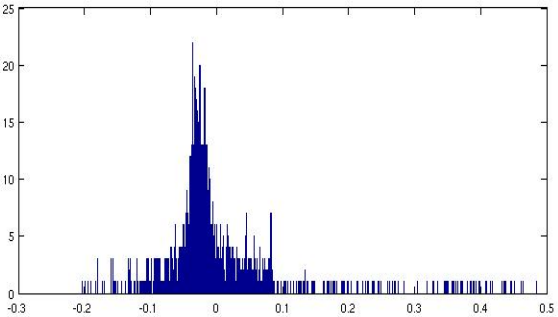


Figure 124: Best solution; Max-Correntropy error pdf for Mean-Square error trained auto-encoder.
Known sensor $P_{I2}$

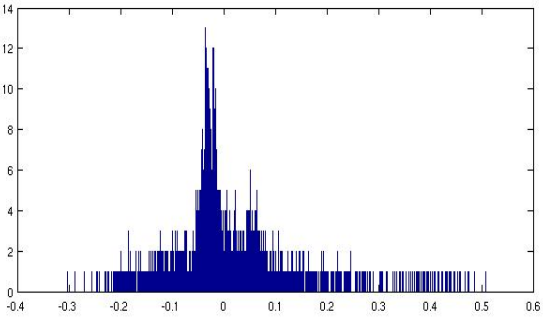


Figure 125: Worst Solution; Min-Square error pdf for Min-Square error trained auto-encoder.
Known sensor $P_{I2}$

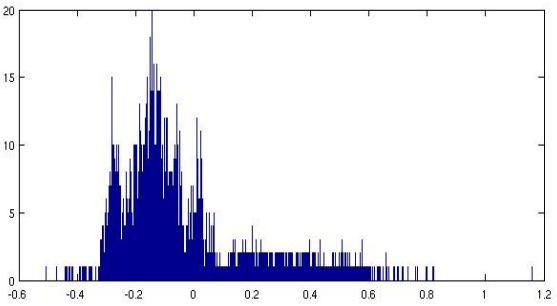


Figure 126: Best solution; Max-Correntropy error pdf for Max-Correntropy trained auto-encoder.
Known sensor $S$

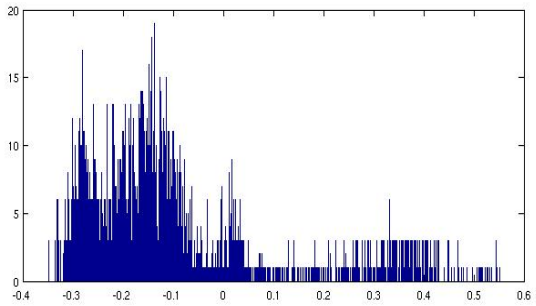


Figure 127: Worst Solution; Max-Correntropy error pdf for Min-Square error trained auto-encoder.
Known sensor $S$

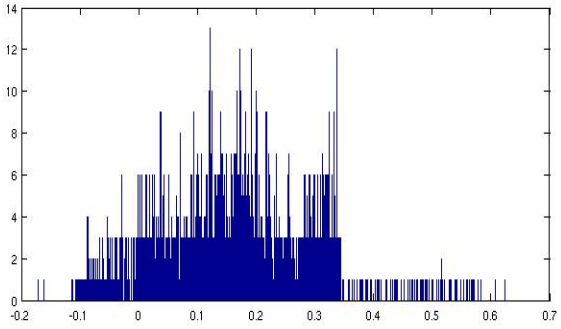


Figure 128: Best solution; Max-Correntropy error pdf for Max-Correntropy trained auto-encoder.
Known sensor *D*

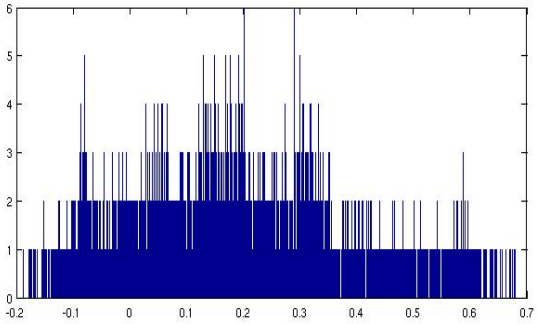


Figure 129: Worst Solution; Min-Square error pdf for Max-Correntropy error trained auto-encoder.
Known sensor *D*

## 5.8 Conclusion and discussion

Comparing the wide spectrum of solutions given by each supervised learning method, one must face the evidences and conclude for the error's correntropy maximization algorithm superiority.

In fact, from the point of view of the auto-encoder's mapping quality independence, error's correntropy maximization proven it's superiority by providing the best solutions for all known and unknown sensors.

The solutions obtained maximizing error correntropy considerably improved error probability density distribution functions, maximizing probability density around the origin for either known and unknown sensors.

| | Max-Correntropy restoration Max-Correntropy error trained encoders | | Min-Square error restoration Max-Correntropy error trained encoders | | Max-Correntropy restoration Min-Square error trained encoders | | Min-Square error restoration Min-Square error trained encoders | |
|---|---|---|---|---|---|---|---|---|
| Sensor | min(ε) | max(ε) | min(ε) | max(ε) | min(ε) | max(ε) | min(ε) | max(ε) |
| P' | -0.2468 | 0.5976 | -0.2468 | 0.5969 | -0.1408 | 0.5328 | -0.3598 | 0.5704 |
| R | -0.2348 | 0.5258 | -0.2178 | 0.5257 | -0.2396 | 0.4671 | -0.3391 | 0.4884 |
| $P_1$ | -0.2793 | 0.5787 | -0.2781 | 0.5808 | -0.1649 | 0.4755 | -0.3863 | 0.5084 |
| $P_2$ | -0.2605 | 0.5348 | -0.2626 | 0.5327 | -0.2673 | 0.5348 | -0.3865 | 0.6032 |
| $P_3$ | -0.3036 | 0.6253 | -0.3059 | 0.6278 | -0.2275 | 0.5840 | -0.3977 | 0.6219 |
| $P_4$ | -0.2140 | 0.6327 | -0.1975 | 0.6334 | -0.3189 | 0.5630 | -0.4161 | 0.5757 |
| $P_5$ | -0.2439 | 0.5365 | -0.2432 | 0.5374 | -0.2352 | 0.4868 | -0.3810 | 0.4830 |
| $P_6$ | -0.2704 | 0.5222 | -0.2697 | 0.5330 | -0.2300 | 0.5222 | -0.2700 | 0.5329 |
| $P_7$ | -0.2239 | 0.6295 | -0.2229 | 0.6280 | -0.1765 | 0.5943 | -0.3854 | 0.6054 |
| $P_8$ | -0.3306 | 0.6286 | -0.3282 | 0.6282 | -0.3106 | 0.4880 | -0.3386 | 0.5342 |
| $P_9$ | -0.2322 | 0.5503 | -0.2156 | 0.5481 | -0.2062 | 0.5176 | -0.4268 | 0.6229 |
| $P_{10}$ | -0.2388 | 0.6028 | -0.2405 | 0.6019 | -0.3860 | 0.5237 | -0.3869 | 0.5574 |
| $P_{11}$ | -0.2396 | 0.5606 | -0.2183 | 0.5760 | -0.2362 | 0.5606 | -0.3111 | 0.5760 |
| $P_{12}$ | -0.3042 | 0.5555 | -0.3033 | 0.5555 | -0.2034 | 0.4854 | -0.3027 | 0.5075 |
| S | -0.5079 | 1.1595 | -0.5078 | 1.1716 | -0.3477 | 0.5515 | -0.4562 | 0.6020 |
| D | -0.1885 | 0.6792 | -0.1885 | 0.6789 | -0.1719 | 0.6246 | -0.2675 | 0.6233 |

**Table 4: Limits for the absolute error for error correntropy maximization and mean square error minimization based learnings.**

However, such maximization has its costs. Compression of the error particles around the origin excluded some operating points which neither algorithms were able to conveniently recover.

One must refer that such sample exclusion isn't frequent, whereas less of 5% of the total number of samples are excluded, for both training algorithms. The problem associated to these spurious results is its abnormal error, sometimes a thousand times bigger than the average of the remaining 95% of the samples, fact which contributes to inflate quite considerably the inferior and superior limits of the error's pdf kernel and respective samples set mean square error, when compared with initial corrupted signals distributions.

This is also verifiable for mean square error minimization method with the major fall back that this learning method does not maximize error probability density around the origin.

| Sensor | Max-Correntropy restoration Max-Correntropy error trained encoders | Min-Square error restoration Max-Correntropy error trained encoders | Max-Correntropy restoration Min-Square error trained encoders | Min-Square error restoration Min-Square error trained encoders |
|---|---|---|---|---|
| P' | 0.0188 | 0.0200 | 0.0041 | 0.0092 |
| R | 0.0141 | 0.0148 | 0.0024 | 0.0083 |
| $P_1$ | 0.0208 | 0.0221 | 0.0030 | 0.0123 |
| $P_2$ | 0.0142 | 0.0158 | 0.0032 | 0.0154 |
| $P_3$ | 0.0180 | 0.0199 | 0.0043 | 0.0148 |
| $P_4$ | 0.0321 | 0.0338 | 0.0047 | 0.0138 |
| $P_5$ | 0.0156 | 0.0161 | 0.0029 | 0.0124 |
| $P_6$ | 0.0116 | 0.0127 | 0.0027 | 0.0073 |
| $P_7$ | 0.0205 | 0.0221 | 0.0040 | 0.0144 |
| $P_8$ | 0.0207 | 0.0219 | 0.0030 | 0.0085 |
| $P_9$ | 0.0182 | 0.0198 | 0.0037 | 0.0130 |
| $P_{10}$ | 0.0224 | 0.0240 | 0.0040 | 0.0107 |
| $P_{11}$ | 0.0206 | 0.0221 | 0.0038 | 0.0104 |
| $P_{12}$ | 0.0193 | 0.0202 | 0.0038 | 0.0081 |
| S | 0.0532 | 0.0536 | 0.0467 | 0.0475 |
| D | 0.0602 | 0.0627 | 0.0391 | 0.0392 |

**Table 5: Max-correntropy and mean square error trained auto-encoders per-sensor output mean square error.**

Do not forget that the data contained in these tables was calculated for normalized inputs. If needed, review normalized inputs individual resolution registered on table 3.

One may add that the verified spurious training answers were easily detected by inspection, due to its incoherency with other known sensor inputs.

Several explanations can be given for the existence of some recovered signals oddness. Between other factors, this is surely also due to imperfect correlation establishment between inputs at defined operating points, once our auto-encoders aren't perfect ones.

Take for example an auto-encoder mapping like the one shown in figure 107, where the neuron's input $I_j$ has little or no influence in respective auto-encoders output $O_j$ outcome, but at the same time alters significantly a set of random output neurons $O_r$. Now, suppose that this input neuron is one of the missing sensors. Once no clue is given to what are the unknown sensors, both training algorithms would wrongly detect more than one bad reading $O_j$, contributing to targeted original signal corruption.

This demonstrates operating point dependency of the auto-encoder's mapper, a constrained search based signal restoration problem refered in previous chapter 5.1. Therefore, known constrained search algorithms still need sufficient information to correctly restore signals.

Even though, general appreciation of the recovered per-sensor error characteristics, namely, the probability density function, mean square error and error kernel limits, reveals undisputed superior quality of the solutions whenever one selects the error correntropy maximization criterion for training method.

Let us now advance to auto-encoder training type solution influence in the present case study.

The best auto-encoder tuning for recovering signal process were consistently the minimized mean square error ones, but only for most part of the input sensors and definitely only when such recover is done by correntropy maximization.

It is important to notice that exceptions to this rule were found exactly for the specially hard to encode wind speed, *S*, and direction, *D,* signals. These signals encoding obstacles derive directly from its large outliers and apparent chaotic behaviour, exemplified through figure 5. For this arduous cases, max-correntropy auto-encoders clearly outclass min-square error ones.

Another interesting fact is the similarity of the pdf's obtained for both restoration supervised learning methods, when using a max-correntropy tuned auto-encoder.

Notice that this probabilistic similarity verifies for all evaluation criteria. In fact, maximums and minimums obtained for each method don't differ more than 1,7% and 7,7%, respectively, mean square error has a maximum difference of 3,9% and histogram shape and maximum density completes the envelope.

This similarity seems to underlie an unique operating point for convergence.

If this could be proven, one could state that for a particular max-correntropy auto-encoder the best solution would always be found independently of the supervised learning method, just as long as the selected method tends to minimize the error between inputs and respective outputs.

Unfortunately, finding compelling proof of such assumption was never an objective of this thesis, leaving this problem open for future works.

Weighting advantages and disadvantages for each supervised learning algorithm obligates us not to forget the worst registered performances attributed to mean square error minimization signal restoration using trained auto-encoders by the same method.

As a matter of fact, if one had singly the mean square error minimization tool to perform signal restoration and auto-encoder tuning, we could only get the worst case scenario possible for all training sets, as error probability density function characteristics here portrayed for this learning method undoubtedly exhibits.

Again, max-correntropy trained auto-encoders and missing signal restoration qualities substantiate this supervised learning algorithm overall preponderance.

# Chapter 6

## Conclusion

# 6 Conclusion

Once studied and evaluated the performance of a theoretic information entropy based criterion, namely error correntropy maximization, which was compared with the well known mean squared error method, when applied to auto-encoder training and missing signal restoration, one can conclude that all pursued objectives of this dissertation were achieved.

Information theoretic learning was reviewed, without forgetting Shannon's and Renyi's information entropy base definitions.

In fact, it's Reniy's quadratic entropy $H_{R2}$ that is estimated recurring to Parzen windows, making possible the consequent definition of information potential, as depicted in chapter 2.2.2.

And it is clear that correntropy, the information theoretic learning selected criterion to accomplish with aimed objectives, is a metric extracted from the information potential $V$ as defined by Xu in (2). It is interesting to remind that information potential based training minimizes the mutual error distances by maximizing quadratic information potential:

$$max(V)=\int_{-\infty}^{+\infty} f_X(x)^2 dx=\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} G(a(i)-a(j),2\sigma^2)$$

Error correntropy minimization aims to minimize the distance between each sample and the origin, therefore maximizing estimated probability density function around the origin:

$$\max_{w,b}\left(J(w,b)\right)=\max_{w,b}\left(V_\sigma(\varepsilon)\right)=\max_{w,b}\left(\frac{1}{N}\sum_{j=1}^{N} k_\sigma(-\varepsilon_j)\right)$$

The first evident advantage of this correntropy criterion is the demand for less computational effort, since complexity of the calculations grow linearly with the number of information particles, or data samples, against quadratic growth when maximizing information potential.

Other advantage is related to the previous knowledge that information potential maximization leads to biased probability density estimation. As one can easily conclude through inspection of the objective function defined to perform error correntropy maximization, the maximum value of this function is at the origin:

$$\max_{w,b}\left(J(w,b)\right)=\min_{w,b}\left[-\left(\frac{1}{N}\sum_{j=1}^{N} k_\sigma(-\varepsilon_j)\right)\right]\Leftrightarrow$$
$$\Leftrightarrow\frac{d}{d\varepsilon_j}\left[-\left(\frac{1}{N}\sum_{j=1}^{N} k_\sigma(-\varepsilon_j)\right)\right]=0$$

Therefore:

$$\frac{d}{d\varepsilon_j}\left[-\left(\frac{1}{N}\sum_{j=1}^{N} k_\sigma(-\varepsilon_j)\right)\right]=0\Leftrightarrow\frac{1}{N}\sum_{j=1}^{N}\left(\frac{d}{d\varepsilon_j}k_\sigma(-\varepsilon_j)\right)=0$$

Once this maximum is set independently for each information particle, one can drop the summation and finally, the maximum will be found at:

$$\frac{d}{d\varepsilon_j} k_\sigma(-\varepsilon_j) = 0 \Leftrightarrow \varepsilon_j e^{-\left(\frac{\varepsilon_j^2}{2\sigma^2}\right)^2} = 0 \Leftrightarrow \varepsilon_j = 0$$

This shows unbiased error probability density estimation, since maximization solution will always be located at the origin.

To show probable biased distribution when using information potential, one may follow the same process to conclude in conformance to what has been herein reported:

$$IF_{a(i)} = \frac{dV}{da(i)} = -\left(\frac{1}{N^2\sigma^2}\right)\sum_{j=1}^{N}\sum_{j=1}^{N} G\left(a(i)-a(j), 2\sigma^2\right)\left(a(i)-a(j)\right) = 0$$

But now, the calculation of the information force for a given particle must include all forces impinged by all the other particles in the data set:

$$\sum_{j=1}^{N} G\left(a(i)-a(j), 2\sigma^2\right)\left(a(i)-a(j)\right) = 0$$

As one can see, we now have a family of possible solutions which satisfies this equation, including the only solution which would be equivalent to the one naturally aimed by correntropy criterion:

$$a(i)-a(j) = 0 \Leftrightarrow a(i) = a(j) = 0$$

Although this is one of the possible family solutions, this would have to be true for all data samples and one can easily conceive this would only happen under very specific conditions.

These are the reasons why this work adopted information correntropy maximization to perform a thorough evaluation of it's properties.

Single hidden layer auto-encoders were chosen to perform the mapping function which allowed the application of both criteria to pursue its solutions. This selection is due to the auto-encoder auto-associative behaviour, which is expressed by it's ability to correlate apparently decorrelated or chaotic information particles, storing information about the encoded phenomenon in its weights and biases.

In fact, auto-encoder effectively works as a phenomenon knowledge base with information compression.

Remember compression is accomplished by hidden layer and output connecting synapses information storage, according to what was described in precedent chapter 4. If enough information is stored in the auto-encoder's mapping function, it is possible to restore loss or erroneous signals.

Some correlation must exist between data samples to allow signal restoration, hence some correct information must integrate the input set which contains the corrupted signals.

To conveniently evaluate the performance of correntropy based criterion, an engineering real

world application was presented in chapters 4 and 5.

The obtained group of solutions for each of the training criteria, as depicted in chapters 4 and 5, were thoroughly evaluated under the following criteria: error probability density function shape, bandwidth and mean square error.

The analysis under such criteria state the overwhelming superiority of the correntropy maximization criterion.

Surprisingly, it has beaten the control method in it's own ground, by finding smaller mean square errors, for the great majority of all the training sets.

From the point of view of engineering, the ability of the maximized error correntropy auto-encoders to encode signals with errors far below signal maximum resolution, which can be seen as zero error measure, allied to compression goals accomplishment, proven that these auto-encoders are an effective way to store information about some phenomenon.

Results also show that maximized error correntropy auto-encoders are in every aspect superior to minimized mean square error ones and signal restoration by means of a max-correntropy criterion compellingly proven its overall superiority under the specified criteria.

In conclusion, max-correntropy criterion usage to perform missing signal restoration and signal encoding expressed it's undoubtful superiority, either when in presence of signals with Gaussian behaviour, like most of the power readings present, either when higher order statistic metrics are present, like the cases of the under-sampled wind speed and direction.

# REFERENCES


(1) Benjamin B. Thompson, Robert J. Marks II and Mohamed A. El-Sharkawi, “On the contractive nature of autoencoders: Application to missing sensor restoration”, Neural networks, 2003, Proceedings of the international joint conference on, pp. 3011-3016, vol. 4.

(2) Dongxin Xu, “Energy, entropy and information potential for neural computation”, Ph.d Dissertation, University of Florida, 1999.

(3) C. E. Shannon, “A Mathematical Theory of Communication,” Bell System Technical Journal, Vol.27, pp.379-423, pp.623-653, 1948.

(4) W. Liu, P. Pokharel, J. Principe, “Correntropy: properties and applications in non-Gaussian signal processing”, Accepted in IEEE Transactions on Signal Processing, 2007.

(5) J.N. Kapur, “Measures of Information and Their Applications”, John Wiley & Sons, New York, 1994.

(6) J. W. Fisher, “Nonlinear Extensions to the Minimum Average Correlation Energy Filter”, Ph.D dissertation, Department of Electrical and Computer Engineering, University of Florida, Gainesville, 1997.

(7) ] D. Erdogmus and J. C. Principe, “Generalized Information Potential Criterion for Adaptive System Training”, IEEE Transactions on Neural Networks, vol. 13, no. 5, September 2002, pp. 1035-1044.

(8) Bessa, Ricardo and Miranda, Vladimiro and Gama, João, “Improvement in Wind Power Forecasting Based on Information Entropy-Related Concepts”, Proceedings of the 2008 IEEE Power and Energy Society General Meeting, July 2008.

(9) I. Santamaria, P. P. Pokharel, J. C. Principe, “Generalized correlation function: definition, properties and application to blind equalization”, IEEE Trans. Signal Processing, Vol. 54, No. 6, pp 2187-2197, June 2006.

(10) Bessa, R., Miranda, V., Gama, J., “Wind Power Forecasting with Entropy-Based Criteria Algorithms”, Probabilistic Methods Applied to Power Systems, 2008. Proceedings of the 10th International Conference on, Volume , Issue , 25-29 May 2008, pp. 1-7.

(11) D. E. Rumelhart, G. E. Hinton and R. J. Williams, “Learning Representations of Back-Propagation Errors”, Nature (London), Vol.323, pp.533-536, 1986.

(12) B. W. Silverman, “Density Estimation For Statistics and Data Analysis”, Chapman and Hall, New York, 1986.

(13) Naylor, A. W., and G. R. Sell, “Linear operator theory in engineering and science”, Springer, New York City, 1982.

(14) Luenberger, D. G., “Optimization by vector space methods”, Jonh Wiley & Sons, April 1997.

(15) Narayanan, S., R. J. Marks II, J. L. Vian, J. J. Choi, M. A. El-Sharkawi & B. B. Thompson, “Set constraint discovery: Missing sensor data restoration using auto-associative regression machines”, Neural Networks, Proceedings of the 2002 International Joint Conference on, 2002 IEEE World Congress on Computational Intelligence, May 12-17, 2002, pp. 2872-2877.