Learning and Nonlinear Models - Journal of the Brazilian Society on Computational Intelligence (SBIC), Vol. 19, Iss. 1, pp. 55-65, 2021

From MSE to Correntropy, a friendly survey

Allan de Medeiros Martins
Federal University of Rio Grande do Norte
Department of Electrical Engineering
allan@dee.ufrn.br

Resumo – Correntropia é uma métrica que vem sendo muito utilizada no lugar do erro médio quadrático em problemas onde se pretente minimizar a divergência entre dados e modelos. Em especial, está em foco atualmente o aprendizado de máquina onde modelos cada vez mais complexos necessitam de dados cada vez mais estatisticamente heterogêneos. Neste artigo iremos dar uma introdução à Correntropia de uma maneira amigável e intuitiva. Ao contrário dos apanhados puramente técnicos, tentaremos balancear uma linguagem precisamente técnica com um texto mais livre e informal. Iremos apresentar a história de como a correntropia chegou a ser desenvolvida, de maneira a levar o leitor a uma sequência temporal coerente e que facilitará o entendimento preciso desta nova métrica.

Abstract – Correntropy is a metric that has been widely used in place of the root mean square error in problems where it is intended to minimize the divergence between data and models. In particular, machine learning is currently in focus, where increasingly complex models require increasingly statistically heterogeneous data. In this article we will give an introduction to correntropy in a friendly and intuitive way. Contrary to purely technical summaries, we will try to balance a precisely technical language with a freer and more informal text. We will present the history of how correntropy came to be developed, in order to lead the reader to a coherent temporal sequence that will facilitate the precise understanding of this new metric.

Keywords – Correntropy, MSE, regression, entropy, information potential.

1 INTRODUÇÃO

Machine learning is a forever growing topic. Applications are overwhelmingly abundant. In this paper we will talk about the learning process itself. In particular, about *Information Theoretic Learning* (ITL in short). To explain what this is, let's explain what learning means and what is the conventional way to do it. Basically, when someone refers to "machine learning" we need to understand what "machine" and "learning" are. To illustrate that, we will focus on a simple example that can be extended to far more complex cases.

Let's consider a simple regression problem. In this problem, we have several pairs of measurements x_i and y_i (i = 1, 2, 3, $\ldots, N$) coming from some experiment or observed relation. The idea is that y depends on x following some rule (also called *model*). In this case, we will consider a super simple model given by

$$y = a\,x + b. \tag{1}$$

The goal is to find the values of a and b that agree with the measurements x_i and y_i that we have. We call this process "fitting the model". In general, we have a more complex model represented as $y = f(x, \mathbf{w})$, where $f(., \mathbf{w})$ is some function parameterized by $\mathbf{w}$. In the case of our simple model, $\mathbf{w} = [a, b]$ and f is the relation showed above. The Figure 1 shows some plots related to this particular regression. In this figure, each line correspond to a certain pair of values of a and b. Each one is the result of a regression. Clearly, the red model is the best model of them all. That means that, the values of $a = 3$ and $b = -1$ are the values that most probably originated the measurements (apart of some random noise).

How to find those correct values of a and b? If the measurements were perfect, all the (x_i, y_i) pairs would fall exactly in the model line and we would only need two measurements to find a and b (why?). Since we have this noisy measurements, we need an algorithm to find the correct values of a and b.

That's it. Going back to machine learning, the *machine* is the model and *learn* is the process of finding a and b. It's as simple as that. The concept of machine learning is extremely simple. What makes it an universe of theory and complexity are the different "machines" and "learning algorithms" that exists.

Going back to ITL, it is a "class" of algorithms that can be used to find the parameters of a model (in another words, to make a machine learn). Before diving into it, let me talk a bit about the conventional way that those algorithms work.

2 MSE

Our problem is to find a good model. What would be a suitable way to measure how good a model is? The first thing that people did (Gauss, I believe) was to compute the difference between the values of the measurement y_i and what the model

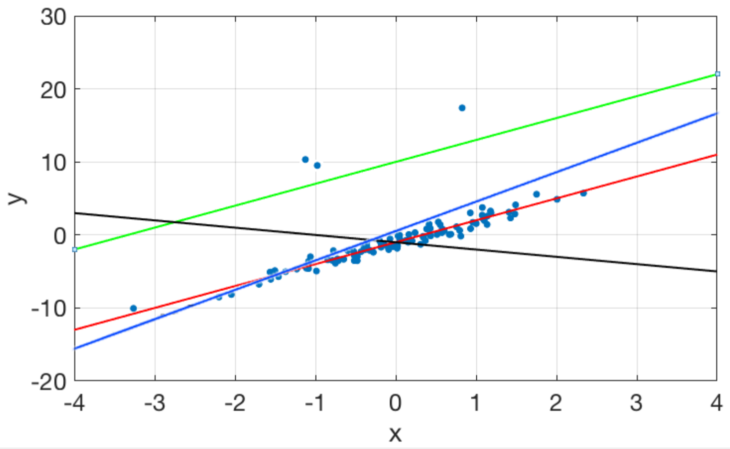

Figura 1: Plot of the measurements (blue dots) and some models (red, green, black and blue lines corresponding to values of a and b of $a = 3$, $b = -1$ for the red line, $a = 3$, $b = 10$ for the green, $a = -1$, $b = -1$ for the black and $a = 3.05$ and $b = -1.04$ for the blue)

predicts based on its respective x_i ($\hat{y}_i = f(x_i, \mathbf{w})$). Hence, for each point, we have an error $e_i = y_i - \hat{y}_i$. The following figure illustrates this concept.

Now, depending on what you compute with those errors you have a different algorithm or class of algorithms. This computation that measures how good the model is, is called *cost function* (called here, J). By far, the oldest and simplest thing you can do is to simple sum all the squares of the errors as follows (why square?):

$$J(\mathbf{w}) = \sum_{i=1}^{N} e_i^2. \tag{2}$$

Notice that J depend on $\mathbf{w}$. In the Figure 2 above, each regression have its own a and b which, in essence, is one specific $\mathbf{w}$. Now, what you want to do is to "optimize" J with respect to $\mathbf{w}$. In this case, optimize means pick the $\mathbf{w}$ that produces the minimum J. This was what people did (and still do) and is the famous **mean squared error** (MSE) criterion.

Hence, MSE is a "class" of algorithms. Depending on how you minimize J, you have a specific algorithm. Those algorithms can differ in several aspects like be interactive, analytic, recursive, etc. In this paper we are not going to talk about this class of algorithms, instead, we will talk about another class: the "entropy" based algorithms.

3 Entropy

Before explain how entropy based algorithms works, let me talk about *why* we would want to use entropy. we will assume that you know a bit of probability (as I was assuming, up to now, that you know calculus). Also, we will try to talk about some specifics aspects of the use of entropy (like outlier rejection). Please keep in mind that there is much more to it.

First, let's refer back to the example in Figure 1. For each data point we have an error e_i associated with a specific regression. Since the data has a random part (the noise) the error is, itself, a random variable. So, for the example of Figure 1, we would have four random variables, one for each model (given the same data). We see that each one have different statistical characteristics. For example, for the green model, we would have large values as being more probable and a very low probability of having small values. As for the black model, we would have a larger "spread" since low errors occur with a certain frequency (when the data is close to the model) but larger values are also present. So, if we have access to the different distributions for each model, we would see something like Figure 3

Now, the sense of "what is best" changes a little. Looking to those plots on Figures 1 and 3, what is the one that represents the best model? In the MSE case, the answer is clear: The best model is the one with less mean squared error. However, it turns out that, for those plots, the one with the least mean squared error is the one with last second order statistics (that is a derivation for another paper) and that's the blue one, not the red !!!

Hence, let's think in terms of an ideal world. What would you expect as a perfect distribution for the error? Wouldn't it be great if the model produced a delta function at zero as error distribution? It would mean that ALL the errors are zero with 100% chance! Of course you can't produce that because the data is already there, and it has some finite "spread". If you are a careful reader, you might have noticed that I'm avoiding to use the word "variance" like in "data with less variance". That's because a distribution can be "spread" in different ways. When we say "variance", we, in general, mean the statistical measure and not "how spread" the data is. But that does not mean that the data is not "concentrated". It might be concentrated in only two points. Two deltas separated by, say 10.0 units in the error axis, have a large variance (statistically speaking) but the data is very concentrated at those two points.

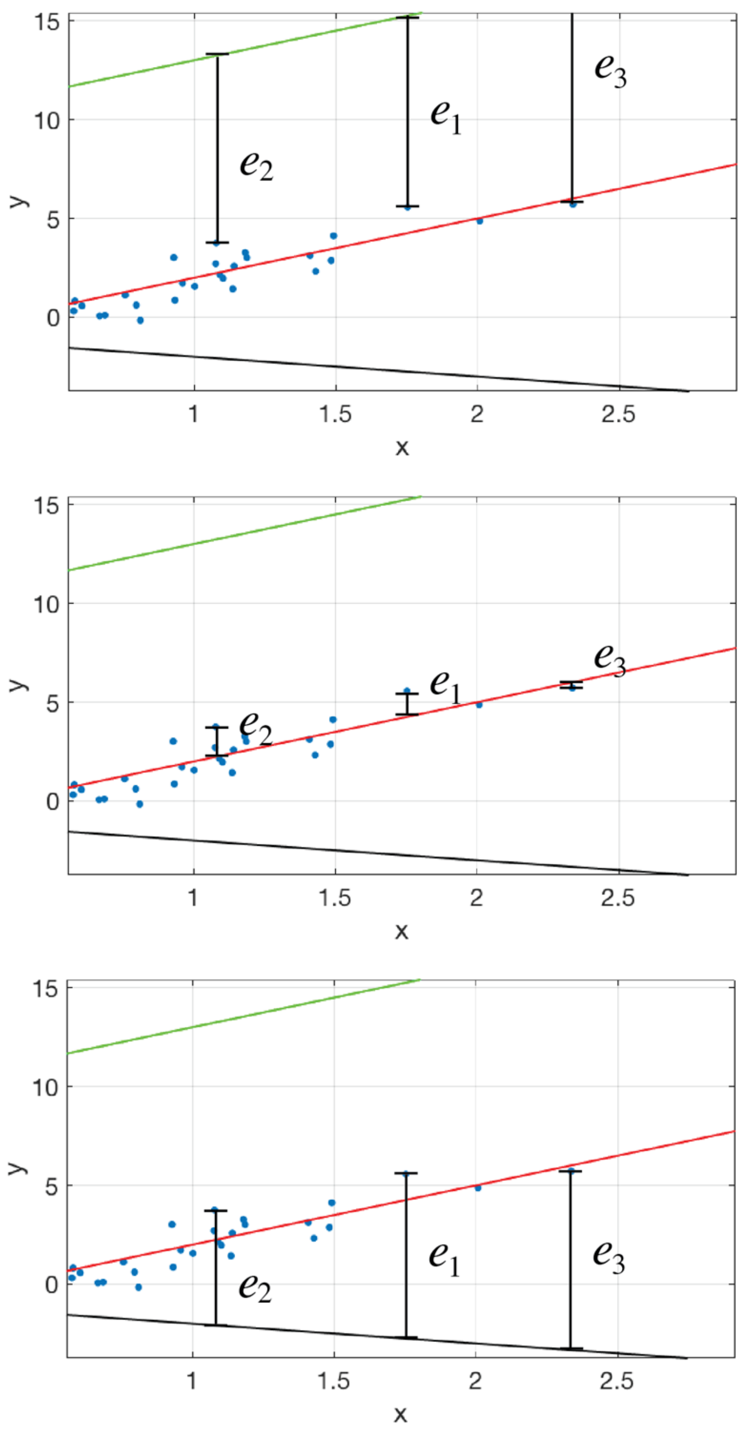

Figura 2: Error between model and data for the first 3 points and some models of figure 1. In reality, we consider the distance from the model to all points, not just the 3 first.

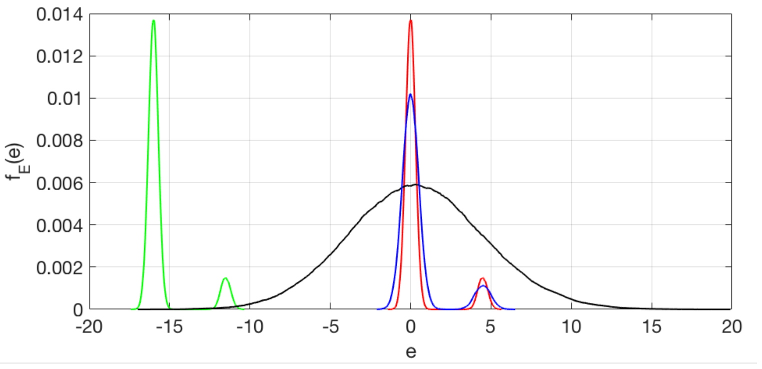

Figura 3: Probability distributions of the error for each data and models showed in figure 2.

So, what would be a good measure for this "concentrationness" or "spreadness" of the data in each model? The answer is: Entropy!

In the plots of Figure 3, the red is the one that presents less entropy. So, problem solved right? Instead of measuring the mean square of the error, we measure entropy of the error and we're good. Well, almost... Yes, if we measure the entropy and minimize it, we have the distributions for the errors as close to a delta as the data allows. The problem is that the green model has the same entropy as the red one. Unfortunately, mathematically speaking, we can't avoid this problem. Entropy is insensitive to changes in the mean of the data, or insensitive to bias. But fortunately, that's pretty much the only fundamental problem with entropy! So, in principle, you can minimize the entropy and then add a bias term (a constant to your model) to make it precise. This is not difficult to do. For instance, in our regression example, you minimize the entropy and get the green model (or any displaced model $f_H(x)$). You pass the data through this "trained" model and compute the mean of the output and call it b_H. Now you compute the mean of the original output and call it b_0. Your final model will be $f_H(x) - b_H + b_0$. That's it.

Well, problem solved! Right? Well, almost again...

Now the problem is: how do we compute the entropy of a set of points? The most known formula to compute entropy is the famous Shannon Entropy, given by

$$H(X) = -\int_{-\infty}^{\infty} f_X(x) \log\left(f_X(x)\right) dx, \tag{3}$$

or, if the variable is discrete, we have

$$H(X) = -\sum_{i=0}^{N} p_X(x_i) \log\left(p_X(x_i)\right). \tag{4}$$

In both cases, the limits of integration and summation must be in accordance to the problem. For instance, if you have two variables, you have a double summation, integration, etc. The big problem is that, either case, we do not have the probability density ($f_X(x)$) or distribution ($p_X(x_i)$).

To solve this problem we turn to estimators. The simplest of those estimators is the histogram. The histogram has a major drawback for us. Remember that the errors depend on the parameters of the model $\mathbf{w}$, therefore, the entropy, in our case, is an algebraic expression. To compute the histogram, we need the numerical values of the errors. But remember that we have numerical values only for the **data**, not for the errors. The errors depend on the parameters of the model which is exactly what we want to minimize over. So, we can't differentiate the histogram with respect to $\mathbf{w}$ to minimize it. Besides, the histogram does not even have an analytical expression to be differentiated in general.

The solution then is to use an analytical estimator, like Parzen windows. In short, the Parzen estimator for the density of a random variable, give some realizations of it, is given by

$$f_X(x) = \frac{1}{N}\sum_{i=0}^{N} k_\sigma(x - x_i). \tag{5}$$

The fundamental item here is the *kernel* used in the estimation, $k_\sigma(x - x_i)$. It is simply a "mini" probability distribution with "width" σ. Without going into details, the most commonly used kernel is a Gaussian with variance σ^2:

$$k_\sigma(x) = G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}. \tag{6}$$

Hence, in our case, the samples of the error are given by (notice that each one depends on $\mathbf{w}$)

$$e_i = y_i - f(x_i, \mathbf{w}), \tag{7}$$

and the density for the error random variable is (notice also that, since e_i depends on $\mathbf{w}$, the final density also depends on $\mathbf{w}$):

$$f_E(e) = \frac{1}{N}\sum_{i=0}^{N} G_\sigma(e - e_i). \tag{8}$$

Good, so we plug this expression in the formula for the entropy, integrate, differentiate with respect to $\mathbf{w}$ and voilà! We have our algorithm (gradient, fix point, etc). Right? Almost yet again...

Just for didactic purpose, let me write how would the final entropy expression look like. Using the Shannon entropy and the Parzen window estimator for the density we have

$$H(E;\mathbf{W}) = \int_{-\infty}^{\infty} \frac{1}{N}\sum_{i=0}^{N} G_\sigma\left(e - e_i\right) \log\left(\frac{1}{N}\sum_{i=0}^{N} G_\sigma\left(e - e_i\right)\right) de. \tag{9}$$

The problem is that this integral in e is impossible to solve! The logarithm of the sum of exponentials has no property to help the solution. They are nasty little math monsters.

Now we do the next big trick! Lets recapitulate why we wanted to use entropy. We want to use entropy because it has this nice property that it minimizes when the distributions are very narrow. But Shannon is *one* of such measures that has this property. There are lots of other options. In particular, we have Renyi's entropy, given by

$$H_\alpha(X) = \frac{1}{1-\alpha}\log\left(\int_{-\infty}^{\infty} f_X^\alpha(x)dx\right). \tag{10}$$

This entropy has the same properties for all α. In particular, if we put $\alpha = 2$ (why not 1?) we have (for the error entropy)

$$H_2(E;\mathbf{w}) = -\log\left(\int_{-\infty}^{\infty}\left(\frac{1}{N}\sum_{i=0}^{N} G_\sigma\left(e - e_i\right)\right)^2 dx\right). \tag{11}$$

Now we can solve the integral (I will let this as an exercise to you) and get

$$H_2(E;\mathbf{w}) = -\log\left(\frac{1}{N^2}\sum_{i=0}^{N}\sum_{j=0}^{N} G_{\sqrt{2}\sigma}\left(e_i - e_j\right)\right). \tag{12}$$

Now we have to stop a bit to contemplate how relevant this expression is. Most people do not realize how important and useful it is. It computes an estimate of the entropy of a random variable, given samples of it! Moreover, the samples can be given algebraically (remember that each e_i depends on $\mathbf{w}$). Now we can estimate the entropy just like we can estimate the mean or variance of a random variable by doing things like $\bar{X} = \frac{1}{N}\sum\limits_{i=0}^{N} x_i$ (by the way, why do we compute the mean like that?).

Finally, we can differentiate $H_2(E,\mathbf{w})$ with respect to $\mathbf{w}$ and we have as many entropy based algorithms as we wish!

4 Information Potential

As we saw in the previous section, we have an estimator for the entropy of the model given some measurement data. Going back to our simple regression example, we can find the a and b that minimizes the entropy by differentiating $H_2(E;a,b)$ with respect to a and b and follow the gradient (for instance).

Notice that we are not interested whatsoever in the value of the minimum entropy. We only want to know the values of a and b that minimizes it. Hence, we can get rid of that nasty logarithm and, instead of minimizing the entropy, we maximize its argument, given by

$$IP(E,\mathbf{w}) = \frac{1}{N^2}\sum_{i=0}^{N}\sum_{j=0}^{N} G_{\sqrt{2}\sigma}\left(e_i - e_j\right). \tag{13}$$

This argument receives a special name: **Information Potential**. The origin of this name is interesting. Entropy is related to the information content of a random variable. If the variable is not very random, its entropy is low. In the case of the estimator we derived, the argument indicates the potential for the variable to be more or less random (depending on the values of the samples e_i). Actually, there is one algorithm that involves moving the samples in the direction of the derivative of the $IP(X)$, effectively generating *information forces*. But that's another paper.

In fact, the information potential is one of the cost functions $J(\mathbf{w})$ for ITL. So, instead of averaging the square of the error, we average all the pairwise combinations weighted by a Gaussian (or the resulted function after integrating the kernel used in the density estimation).

As an example, lets see what happens when we try to use the information potential in our simple regression problem. We plug the expression for the errors using our simple model and get the following

$$\begin{aligned} J(\mathbf{w}) &= \tfrac{1}{N^2}\sum_{i=0}^{N}\sum_{j=0}^{N} G_{\sqrt{2}\sigma}\left(y_i - ax_i - b - y_j + ax_j + b\right), \\ &= \tfrac{1}{N^2}\sum_{i=0}^{N}\sum_{j=0}^{N} G_{\sqrt{2}\sigma}\left(y_i - ax_i - y_j + ax_j\right). \end{aligned} \tag{14}$$

As you can see, the entropy (now our cost function) is independent of the term b. As we explained in the beginning, any shift in the error by a constant would just shift its probability density by the same constant, leaving the entropy unaltered. Because of that, we have to do the trick of biasing the model (which is simple). If we differentiate this expression with respect to a and maximize it (using gradient descent or something like that) we would get a better estimate then using MSE. That solution corresponds to finding models like the red and green in Figures 1 and 3.

We now have the means to use entropy to train models. However, lets analyze two fundamental aspects of this estimator.

4.1 Computational cost

As we can see in the expression of the information potential, we have a double sum. Although it is a symmetric computation, which means that we actually just need to compute half of the terms, the algorithm is still $O(N^2)$ in complexity. Since the sums involve Gaussians, there is a clever trick to compute it using Gaussian transforms. Basically we use the fact that, when the argument of the Gaussian is large, we can ignore it. In summary, the double sum is an issue that must be addressed when using entropy to train a model.

4.2 Kernel size

The *kernel size* is **the** most important aspect of ITL. When we derived the entropy estimator, there was an important piece of information that was overlooked. It was the *width* of the kernels σ used in the estimator. We call it the **kernel size**. In his paper, Parzen showed that, in order for us to have a good estimation of the pdf of the data, the size of the kernel must decrease to zero as the number of points tends to infinity. The exact way this decreasing must occur depends on the data itself. There are some "rules" that tries to estimate this relationship (σ versus N) but in the ITL case, this it is very difficult because, for each model parameter $\mathbf{w}$, we have a different distribution. So, in theory, we should have an estimate for σ that is $\mathbf{w}$ dependent. However, remember that we do not want the value of the entropy or information potential themselves. We rather want the $\mathbf{w}$ that maximizes it. In practice, we perform tests with different values of kernel sizes and, in general, it is not too difficult to find a suitable one. In the end, even if the estimate of the entropy is not perfect, the gain of having *some* estimate surpasses by far the disadvantages of not being precise in the measure.

4.3 Demo

In the address in [7] you can find a little demo for the concept of Information Potential. The first plot is the plot for the regression. You can see the data and two models. The gray model is the correct model. It is the true phenomenon that relates x to y. The data, as you can see, do not fit perfectly on the model due to noise. The second model you can set by changing the sliders "a" and "b". The subsequent plot is the density function of the error for the model. You can set the kernel size for the estimate of the distribution on the third slider. Between the plots we can see the MSE and entropy measurement for blue model.

You can play with the sliders and see how things change. In particular, try to find the positions where you have minimum MSE and/or minimum entropy. Compare the values for a and b in both cases. From the true model, try to compute the correct values (another exercise for you).

5 Correntropy

When we look at the information potential expression, we realize that it looks like some sort of estimation of an expected value of something. In fact, if e_i was a stochastic process, we might imagined that there is some measure between the different lags in the process and we were taking the average of that measure.

That leads us to think that there exists such measure, that we are going to call "correntropy", that measures some sort of statistical relationship between two random variables. In fact, the Information Potential is the expected value of this quantity when the two random variables are samples with different lags in some stochastic process. Since the information potential came from Renyi's Entropy, and now we are measuring some sort or "correlation" between two random variables, the name *Correntropy* seems adequate. Hence, initially, correntropy will be estimated like so

$$\tilde{V}(X,Y) = \frac{1}{N}\sum_{i=1}^{N} k_\sigma(x_i - y_i), \tag{15}$$

and defined as

$$V(X,Y) = E\left\{k_\sigma(X,Y)\right\}. \tag{16}$$

Notice that the kernel used here does not need to be the Gaussian kernel. It came originally from the Parzen estimation of the error probability in the IP. This will be important in a later section. Also, notice the difference in notation. The second expression depends on the random variables directly, while the first depends on its samples (random variables are non-bold uppercase and values are lowercase).

Now we have this weird but simple-looking expression that apparently measures some sort of "correlation" between variables X and Y. In fact, as we are going to see, this measure is far more powerful than a simple correlation. It is, in fact, related to high order statistics and inner products in a larger Hilbert space.

Before diving into the deeper analysis of this measure, let us first point out some important facts. First, we have a bivariate measure. It means that we are measuring some sort of relationship between two sets of samples (realizations of random variables). As opposed to the IP, now we need two random variables to plug into the measure. Second, we jumped from an estimation to a *definition* of a measure. For theoretical purposes, we can (and will) use the definition of correntropy, instead of its estimation, to draw important conclusions about its statistical and analytical behavior. Lastly, by construction, correntropy depends on a kernel and a kernel size. That is important because we frequently see papers claiming that they "generalized" correntropy when, in fact, they exchanged the Gaussian kernel for some other kernel and obtained some more general result.

In the next three sections, we will try to gain some theoretical insight about correntropy. First we will "dissect" it and show that it corresponds to a generalized statistical measure. Then, we will present two fundamental interpretations for correntopy that leads to two different understandings of it. There is no correct interpretation. We simply have different ways of interpreting what the expression can mean.

5.1 Generalized statistics

Here we will dissect the expression for correntropy. The tool we will use for that is the Taylor series expansion. For correntropy, using the Gaussian kernel around $x_i = y_i$ (why this point in particular?):

$$V(X,Y) = \frac{1}{\sqrt{2\pi}\sigma} E\left\{\sum_{i=0}^{\infty} \frac{(-1)^i (X-Y)^{2i}}{i! 2^i \sigma^{2i}}\right\}. \tag{17}$$

There is something very interesting hidden in this expansion. To see what it is, let's look at the first terms in this series (pulling always the constant factors for each one for simplicity sake):

$$\begin{aligned} V(X,Y) &= c_1 E\left\{(X-Y)^2\right\} + c_2 E\left\{(X-Y)^4\right\} + \cdots \\ &= c_1 E\left\{X^2\right\} + c_1 E\left\{Y^2\right\} + 2c_1 E\left\{XY\right\} + c_2 E\left\{X^4\right\} + c_2 E\left\{Y^4\right\} + \cdots \end{aligned} \tag{18}$$

Since we can always whiten our data making all first single variable constant ($E\{X^2\} = 1$, $E\{Y^2\} = 1$, $E\{X^4\} = c$ and etc), we can rewrite the correntropy as something like

$$V(X,Y) = c_0 + \frac{c_1}{\sigma^3} E\left\{XY\right\} + \frac{c_1}{\sigma^6} E\left\{XY^3\right\} + \frac{c_1}{\sigma^6} E\left\{X^2Y^2\right\} + \cdots \tag{19}$$

We can now see that correntropy is a generalized correlation measure that takes into account all the even statistical moments of the variables X and Y. Moreover, if we increase the kernel size σ, higher order terms vanishes "more quickly" than the lower ones. Hence, if we make the kernel size wide enough we can make correntropy behave like a simple second order correlation. Well, there is a little technical drawback on this last property of correntropy. As it turned out, the constant c_0 sitting there in the expansion is proportional to $1/\sigma$, so if we increase the kernel size too much, we get a constant function, not the simple correlation function between X and Y. But that is a technical issue, since we can always remove some value to whatever we are measuring with correntropy.

Now we can use correntropy as a generalized correlation function. Any statistical method that uses correlation now can have a correntropy version. Just like we exchanged MSE by entropy in the previous sections, we can now exchange correlation to correntropy.

5.2 RKHS Interpretation

When dealing with a stochastic process or stochastic function $X(t)$ (or $X[m]$ in the discrete case), it is very common to refer as its covariance "kernel" as

$$R(s,t) = E\left\{X(s)X(t)\right\}. \tag{20}$$

This is an important function that turns out to describe a lot about the process X itself. However, this is the case when the process has second order nature. In other words, when the process is mainly described by the covariance between the "taps" at t and s. Unfortunately this is not always the case.

Nevertheless, Parzen (him again) showed in a brilliant paper that a stochastic process gives rise to an abstract Hilbert Space (technically a Reproducing Kernel Hilbert Space, or RKHS in short). Basically, for a given stochastic process X, there exists a RKHS H_X in which random variables $X(t)$ can be viewed as vectors in this abstract space. Hence, everything you can do with random variables, we can now have an equivalent operation as vectors. For instance, inner products with random variables in one space, means correlation in the other. We can also measure the "square length" of a variable and that means variance. And so on. In essence, Parzen's paper tremendous contribution wasn't to provide new tools to deal with random variables. Instead, he gave us a very powerful interpretation of random variables as vectors in this abstract RKHS spanned by a stochastic process.

As an important note, RKHS became a big trend in the past years as the "kernel method" for machine learning arose. In particular, the theorems due to Moore-Aronszajain and Mercer that show that, when you evaluate the kernel in the "original" space, you are performing a inner product in the "feature" space that have a bigger (even infinite) dimension. Then, due to another big theorem, we know that when we project things from a lower dimension to a higher dimension, the projected vectors tend to be linearly related to each other. Even better, in order for this to be true, you just need the kernel to be a positive definite function. Because of that, we can do all kinds of complex projections using non-linear functions as kernel and we would be, in fact, doing linear algebra in some complex abstract space. That is the famous "kernel trick".

Now if we call "probability space" as "original space" and "Hilbert Space" as "feature space" we will have kernel methods for random variables! What would be the relationship of Parzen's RKHS with correntropy? If we remember the definition of correntopy as the expected value of a function of the two random variables, It so happens that if that function is positive definite (as the Gaussian kernel is, for instance) we have exactly the same "kernel trick" in statistics that we have in machine learning!

In essence, when we compute correntropy, we are computing the inner product in some RKHS spanned by some random process that have X and Y as vectors transformed by a crazy function, like

$$V(X,Y) = E\left\{k_\sigma(X,Y)\right\} = \langle \Phi(X), \Phi(Y) \rangle = E\left\{f(X)f(Y)\right\}. \tag{21}$$

So, in summary, correntropy can be viewed as a mean to compute a linear inner product in some space in which the transformation from the original space is non linear. In another world, we can perform the kernel trick for random variables!

5.2.1 Example

As an example, let's consider a case of non-linear regression with correntropy. The idea is to use the "correntropy kernel trick" to go from a linear regression to its non-linear version. So, let's start with the linear formulation. We have the following relationship:

$$y = \mathbf{w}^t \mathbf{x}. \tag{22}$$

We will suppose that we have samples of y and $\mathbf{x}$ and wish to find $\mathbf{w}$.

The trick is to "kernelize" this problem, but still keep things linear. To do this, we will write $\mathbf{w}$ as a linear combination of some "support" vectors $\mathbf{c}_l, l = 1, 2, \ldots, L$ as

$$\mathbf{w}^t = \sum_{l=1}^{L} a_l \mathbf{c}_l^t. \tag{23}$$

Now, instead of trying to find $\mathbf{w}$, we will choose some vectors $\mathbf{c}_l$ (somewhat arbitrarily) and try to find a_l. If we do this in the linear case, we might think we gain anything since we are still solving a linear system. That is true, but we can keep going and derive something useful later.

We now have some measurements for y as $d_k, k = 1, 2, \ldots, K$ and for $\mathbf{x}$ as $\mathbf{x}_k$. We then substitute the expression for $\mathbf{w}$ and get

$$\begin{aligned} d_k &= \sum_{l=1}^{L} a_l \mathbf{c}_l^t \mathbf{x}, \\ &= \sum_{l=1}^{L} a_l \sum_{i}^{d} c_{i,l} x_{i,k}. \end{aligned} \tag{24}$$

As all the measurements and choices of vectors are random variables, we choose the following notation:

$$D_k = \sum_{l=1}^{L} a_l \sum_{i}^{d} C_{i,l} X_{i,k}. \tag{25}$$

That is the relationship that should provide the pathway to our kernel trick. Let's take the expected value on both sides of the previous equation:

$$E\left\{D_k\right\} = \sum_{l=1}^{L} a_l \sum_{i}^{d} E\left\{C_{i,l} X_{i,k}\right\}, \tag{26}$$

and now we have $E\left\{C_{i,l}X_{i,k}\right\}$, which is a simple linear stochastic kernel. That means that this whole thing can be consider as a linear regression on a complex RKHS that has a non-linear transform back to our original space. Since this expected value is a inner product on this RKHS, we can translate it to our original space via correntropy kernel trick as

$$E\left\{D_k\right\} = \sum_{l=1}^{L} a_l \sum_{i}^{d} V\left(C_{i,l}X_{i,k}\right). \tag{27}$$

Now, we can find the constants a_l via

$$\mathbf{a} = \mathbf{M}^{\perp}\mathbf{p}, \tag{28}$$

where the little cross on $\mathbf{M}^{\perp}$ means the pseudoinverse and the variables are defined as

$$\begin{aligned} \mathbf{M} &= \begin{bmatrix} V\left(C_{1,1}X_{1,1}\right) + \cdots + V\left(C_{d,1}X_{d,1}\right) & \cdots & V\left(C_{1,L}X_{1,1}\right) + \cdots + V\left(C_{d,L}X_{d,1}\right) \\ \vdots & \ddots & \vdots \\ V\left(C_{1,1}X_{1,K}\right) + \cdots + V\left(C_{d,1}X_{d,K}\right) & \cdots & V\left(C_{1,L}X_{1,K}\right) + \cdots + V\left(C_{d,L}X_{d,K}\right) \end{bmatrix}, \\ \mathbf{p} &= \begin{bmatrix} E\left\{D_1\right\} \\ \vdots \\ E\left\{D_K\right\} \end{bmatrix}. \end{aligned} \tag{29}$$

One important aspect of this solution is that we have K random variables for each problem variable ($\mathbf{x}$, $\mathbf{c}$ and $\mathbf{d}$). This means that we should "repeat" the measurements K times in order to obtain enough statistics to be able to compute correntropy for each variable. That is not a problem because we can always divide our data into K groups (separated independently) and use the groups as repeated measurements.

Now we have the relationship that computes y in the original space

$$y = \sum_{l=1}^{L} a_l \sum_{i}^{d} V\left(C_{i,l}X_i\right). \tag{30}$$

In general, we can choose the support vectors as we want, so we pick them as random variables with a delta distribution over some constants $\mathbf{c}_l$. Also, the point $\mathbf{x}$ is an *a priori* measurement, so we know its value and can consider it as a random variable with a delta distribution. If that is the case, the expected value of the kernel is the kernel itself. Hence, we have our non-linear model as

$$y = \sum_{l=1}^{L} a_l \sum_{i}^{d} k_\sigma\left(c_{i,l}x_i\right). \tag{31}$$

5.3 Probability Interpretation

Correntropy hides another super interesting interpretation. This time, let's show this "property" of correntropy by going backwards. Let's first state the property and then arrive at the definition of correntropy. It turns out that correntopy is actually the probability density of the event $X = Y$ when the kernel size tends to zero. So, in essence, when we estimate correntropy between two variables (we say estimate because remember that the definition involves an expected value and in practice the kernel size is never zero) we are estimating the probability that both variables are equal. Moreover, this estimation is numerically the same as the estimation using the Parzen windows method for estimating the density of the joint distribution of X and Y.

So, let's start by estimating the density of the joint space by

$$f_{XY}\left(X, Y\right) = \frac{1}{N}\sum_{i=1}^{N} k_\sigma\left(x - x_i, y - y_i\right). \tag{32}$$

Notice that the joint space is bi-dimensional (X, Y). Because of that, the kernel must also be a bi-dimensional entity. For the sake of simplicity, we will see how this works with a Gaussian kernel, but the concept is more general. The Gaussian kernel, then, is defined as

$$k_\sigma\left(x, y\right) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{33}$$

Now we can compute the probability of the event $X = Y$. To do that, we should integrate the joint density along the curve that defines the event. In the case of $X = Y$, it is very simple and we just need to integrate along the 45 degree line with a delta function. This is achieved by the integral

$$\begin{aligned} P(X=Y) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_{XY}(x,y)\,\delta(x-y)\mathrm{d}x\mathrm{d}y, && (34)\\ &= \int_{-\infty}^{\infty} f_{XY}(u,u)\,\mathrm{d}u. && (35)\end{aligned}$$

Using the Parzen estimate for the density we have

$$\begin{aligned} P(X=Y) &= \frac{1}{N}\sum_{i=1}^{N}\int_{-\infty}^{\infty} k_\sigma(u-x_i,u-y_i)\,\mathrm{d}u, && (36)\\ &= \frac{1}{N}\sum_{i=1}^{N}\int_{-\infty}^{\infty} \frac{1}{2\pi\sigma^2}e^{-\frac{(u-x_i)^2+(u-y_i)^2}{2\sigma^2}}\mathrm{d}u. && (37)\end{aligned}$$

If we solve the integral, we can write the expression for the probability as

$$P(X=Y) = \frac{1}{N}\sum_{i=1}^{N} G_{\sqrt{2}\sigma}(x_i-y_i), \tag{38}$$

which is exactly the estimate for correntropy when we use the approximation for the expected value!

5.3.1 Example

We can use this interpretation of correntopy as a cost function. In fact, as a cost function, it is extremely intuitive because if you have a measured random variable and a parameterized model, the cost function will be probability that the model is equal to the measurement! Hence, you simple maximize this quantity and you're done. As a simple example, let's consider a polynomial regression. Our model now will be

$$y = ax^2 + bx + c. \tag{39}$$

We have some measurements of the pair (x_i, y_i) and, with that, we can compute the estimate of $\hat{y}_i = ax_i^2 + bx_i + c$. Now we have the two random variables Y and $\hat{Y}$. Finally, we just need to maximize the probability of the event $Y = \hat{Y}$. In other words, our cost function (cost here is a mislead word) will be $V(Y, \hat{Y})$ as

$$P(Y=\hat{Y}|a,b,c) = \frac{1}{N}\sum_{i=1}^{N} G_{\sqrt{2}\sigma}\left(ax_i^2+bx_i+c-y_i\right). \tag{40}$$

As you can see, unlike the entropy in the previous sections, correntropy have no bias problem (if you change the bias term, the correntropy does not remains constant) and its estimator has only one summation. The outlier rejection property is still present. One can notice that points that are too far from the model, will be expressed as a large value in the argument of the kernel. That makes it has very little impact on the overall cost function value.

5.4 Demo

In [8] we try to illustrate the probability interpretation of correntropy. It shows the polynomial regression described in the previous section. The sliders control the value of the parameters of the parabola (a, b, c) and the kernel size. The first plot shows the measurements (x_i, y_i), the correct model and chosen model. As you change the sliders, you can see the fitting parabola change as well as the samples in the space $Y \times \hat{Y}$ in the subsequent plot. The third plot shows the probability density estimation for the joint space $Y \times \hat{Y}$ and a plane that cuts the probability at the line $Y = \hat{Y}$. The idea is that, when you fit the right parabola, the area of the joint density will be maximum in the plane $Y = \hat{Y}$.

6 Conclusions

Information Theoretic Learning is a great framework for machine learning. It has a lot more to it than we presented here. The goal is to introduce the reader to this new and awesome concept. Since it deals with entropy, things like mutual information, Cauchy-Schwarz relations, and etc, can be used in the process of learning. Even non-parametric algorithms can benefit from ITL. Because of that, not only it is a new way to define cost functions, but rather it is a unifying framework for learning in general! Depending on how you use the metrics, one can have supervised and non-supervised algorithms, fitting of non-linear and linear models, generalized statistics, among others.

References

[1] J. C. Principe. *Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives*. Springer, 2001.

[2] E. Parzen. "On Estimation of a Probability Density Function and Mode". *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[3] E. Parzen. "Statistical Inference on Time Series by Hilbert Space Methods". *Office of Naval Research Report Number CHEONR23*, vol. I, 1959.

[4] W. Liu, P. P. Pokharel and J. C. Príncipe. "Correntropy: Properties and Applications in Non-Gaussian Signal Processing". *IEEE Transactions on Signal Processing*, vol. 55, no. 11, pp. 5286–5298, 2007.

[5] J.-W. Xu and J. C. P. A. R. C. Paiva, Il Park. "A Reproducing Kernel Hilbert Space Framework for Information-Theoretic Learning". *IEEE Transactions on Signal Processing*, vol. 56, no. 12, pp. 5891–5902, 2008.

[6] D. Erdogmus. "Information Theoretic Learning: Renyi's Entropy and its Applications to Adaptive System Training". Ph.D. thesis, Computational Neuro-Engineering Laboratory, University of Florida, USA, 2000.

[7] A. Martins. "ITL Demo". `https://www.dca.ufrn.br/%7Eallan/blog/wp-content/pages/apps/ITL1/IP.html`, 2008.

[8] A. Martins. "Correntropy Demo". `https://www.dca.ufrn.br/%7Eallan/blog/wp-content/pages/apps/ITL1/correntropy.html`, 2008.