Robust Timeseries Analysis in the Context of Multiple Asynchronous Input Channels

Ryan Michael
kerinin@gmail.com

March 9, 2009

Abstract

Spectral Analysis is a common form of analysing timeseries data. We propose a method of spectral analysis based on sets probability distributions generated from subsets of the observed data. By decomposing the observations into a set of observed behaviors, the spectral analysis can be used more accurately to predict the timseries' behavior.

1 Introduction

1.1 Existing Work

http://en.wikipedia.org/wiki/Linear_discriminant_analysis

1.2 General Overview

The goal is to create a method of statistical inference capable of processing timeseries data which is both multi-variate and exhibits different behaviors at different times. This type of data is common, and developing a robust method of analysis has applications in many domains. The general approach is to create a series of estimates using subsets of the observed data, and to then combine these estimates in an intelligent manner which captures the relevance of each estimate to the current prediction task. By using a set of 'typical' estimates, we are able to reduce the computational demands of the system, as each estimate is a condensed representation of the data from which it was derived. This approach also allows us to reduce data redundancy by only using distinct estimates.

The most basic operation used in this system is the estimation of probability densities. Based on a set of observations drawn from some random process, we generate an estimate of the underlying probability distribution. This estimate tells us the probability of each point in the input space being observed. Areas of the input space in which a dense set of observations are observed are given high probability, while areas of the input space with few observations are given low probability. This

basic operation, in combination with some basic laws of statistics allow us to build the full inference system.

We assume that observations are pulled from multiple independent sources, all of which respond to some underlying phenomena. For instance one set of observations could be from a microphone and another from a light detector. We do not know how the inputs are related or to what extent their behavior changes over time. For example one input could be a steady sine wave and another could be based on the decay of a radioactive isotope. In the former case previous observations of the source are useful, in the latter they're not. Alternately two inputs could be light detectors in the same room or they could be on different continents; in the former their input would be highly similar, in the latter not as much.

Our general strategy has three phases; generating estimates, correlating estimates, and applying estimates to a given prediction task.

1.3 Generating Estimates

Estimates are generated by picking a time window at random and only dealing with observations which take place in that window. Using these observations we create an estimate of the probability distribution underlying the observations (this distribution would include time as a variable). This process is repeated until we feel we have a reasonable sample of the system as a whole, at which time we can start to correlate the estimates.

1.4 Correlating Estimates

The goal of correlating estimates is to determine the relationships between estimates at different times and from different sources. Estimates are correlated by treating them as variables whose value for a given time window is determined by the extent to which the observed data corresponds to the estimate. For each time window there exist a set of estimates which have some value describing their accuracy at predicting the observed value. We can treat the accuracy measurement of each estimate as a multi-dimensional point, each dimension determined by an estimate's accuracy. Using a set of these points taken at different times, we create a probability distribution estimate. The domain of this probability distribution has the same dimensionality as the number of estimates we have generated. This probability density allows us to predict the probability of an estimate in of one source based on the probability of an estimate in of another source because frequent combinations of accuracy values will have higher probability than other combinations. This 'correlation density' also tells us which estimates are most commonly observed - information we can use in conjunction with estimate similarity to determine which estimates to use and which to discard.

1.5 Making Pedictions

Once the estimates have been correlated, we are able to generate predictions. For simplicity, we'll assume that the first two phases occur on as set of 'training' data which is representative of the underlying data, while prediction takes place continuously using new sets of observations which occur in some time window. We make predictions for a given source by combining the existing estimates we have for that source based on their accuracy at predicting the given observations. Because we have determined the correlation between estimate accuracy of different sources, we can use other sources to refine our confidence in each estimate of the given source; the influence of estimates of the given source which do not correspond to a likely 'point' in the correlation density are suppressed while the influence of estimates which correspond to likely points in the correlation density are enhanced.

2 Formal Description

2.1 Problem Setting

We begin with a hidden random variable

$$X = (\Omega, \mathcal{F}, \mathcal{P})$$

Our knowledge of X comes from a set of independant sources which we treat as random variables generated by X:

$$X^n : \Omega \mapsto \Omega^n \in \mathbb{R}^d \times \mathbb{R}_+$$
$$X^n = (\Omega^n, \mathcal{F}^n, \mathcal{P}^n)$$
$$\mathbf{X} = [X^0, ..., X^c]$$

We refer to each of these sources as a channel, and refer to each channel as the i^{th} element of the set $\mathbf{X}$:

$$C^n = [\Omega^n, \mathcal{F}^n, \mathcal{P}^n]$$

For each channel we are given a set of ℓ observations of dimension d, each with a time value:

$$\mathbf{x}_i^n = (x_i^n, t_i^n)$$
$$X^n = [\mathbf{x}_0^n, \ldots, \mathbf{x}_\ell^n] \in \mathbb{R}^d \times \mathbb{R}_+$$

Given a set of time durations $\theta \in \Theta$, we define a set of time widows:

$$\mathcal{T}^n = \begin{bmatrix} (t_0^n, \theta_0) & \dots & (t_0^n, \theta_z) \\ \vdots & \ddots & \vdots \\ (t_\ell^n, \theta_0) & \dots & (t_\ell^n, \theta_z) \end{bmatrix}$$

We assume that the probability of X is a time-dependent mixture of some set of distributions $[f_0, ..., f_w]$ whose influence is unknown and changes over time. We can refer to the weighted mixture which corresponds to a given time window (t, θ) as:

$$\mathcal{P}(t, \theta) = \sum_i \delta_i \cdot f_i \tag{1}$$

where δ_n is the weight corresponding to f_n. Finally, we assume that a similar mixture of densities can be determined for each channel:

$$\mathcal{P}^n(t, \theta) = \sum_i \delta_i^n \cdot f_i^n \tag{2}$$

2.2 Single Channel Setting

We begin by considering the case where only one channel exists, so for now we will omit the superscript and refer to $X^n = (\Omega^n, \mathcal{F}^n, \mathcal{P}^n)$ as $X = (\Omega, \mathcal{F}, \mathcal{P})$. Given a set of test data $\hat{X}$, our goal is to estimate the probability distribution of X over some time window $\mathcal{T}_{\hat{x}}$:

$$\begin{aligned} &\hat{x}_i = (\hat{x}_i, t_i) \\ &\hat{X} = [\hat{x}_0, \dots, \hat{x}_k] \\ &\mathcal{T}_{\hat{x}} = (t_{\hat{x}}, \theta_{\hat{x}}), \quad t_{\hat{x}} < \min_t \hat{X} < \max_t \hat{X} < t_{\hat{x}} + \theta_{\hat{x}} \end{aligned}$$

We begin by defining the subset of the training observations which fall into each time window, scaled and shifted to the interval $t \in [0, 1)$:

$$S_{t,\theta} = \left[\left(x_i, \frac{t_i - t}{\theta} \right) \,\middle|\, \quad x_i \in X,\ t \le t_i < t + \theta \right] \tag{3}$$

$$\mathcal{S} = \begin{bmatrix} S_{t_0,\theta_0} & \dots & S_{t_0,\theta_z} \\ \vdots & \ddots & \vdots \\ S_{t_\ell,\theta_0} & \dots & S_{t_\ell,\theta_z} \end{bmatrix}$$

We treat $\mathcal{S}$ as a random process:

$$\mathcal{S} = [\Omega^{\mathcal{S}}, \mathcal{F}^{\mathcal{S}}, \mathcal{P}^{\mathcal{S}}]$$

and observe that $\hat{X}$ can be treated as an observation of $\mathcal{S}$:

$$S_{\hat{x}} = \left[\left(x_i, \frac{t_i - t_{\hat{x}}}{\theta_{\hat{x}}} \right) \,\middle|\, \quad (x_i, t_i) \in \hat{X} \right] \tag{4}$$

We can now frame the task of estimating the probability distribution of $\hat{X}$ as a task of estimating a probability density function φ for the random process $\mathcal{S}$:

$$\Pr(X = \mathbf{x} \mid S_{\hat{x}}) \mapsto \ \Pr(\mathcal{S} = \{S_{\hat{x}} \cup \mathbf{x}\}) \ \simeq \ \varphi(\mathbf{x}, S) \tag{5}$$

2.3 Multiple Channel Setting

In order to extend this result to settings in which multiple channels exist, we return to 3 and extend the scope to multiple channels:

$$S_{t,\theta} = \left[\left(x_i^n, \frac{t_i - t}{\theta} \right) \Big| \quad x_i^n \in \mathbf{X},\ t \leq t_i < t + \theta \right] \tag{6}$$

In this case, we treat each channel as an orthonormal basis of the abstract space $\Omega^{\mathcal{S}}$. Equation 4 is likewise extended in the same manner:

$$S_{\hat{x}} = \left[\left(x_i^n, \frac{t_i - t_{\hat{x}}}{\theta_{\hat{x}}} \right) \Big| \quad (x_i^n, t_i) \in \hat{\mathbf{X}} \right] \tag{7}$$

3 Parzen Window Estimation

One method of evaluating 5 is by using the Parzen Window method. We will again begin by considering the single-channel case, then extend the resulting equations as necessary

3.1 Single Channel Parzen Window

The Parzen Window method requires the definition of a metric over the abstract space $\Omega^{\mathcal{S}}$. Such a metric can be defined using the symmetric Kullbeck Liebler divergence with probability measures $\phi_n(\mathbf{x}) \simeq Pr(X \mid S_n)$:

$$\begin{aligned} D_{KL}(S_n \| S_m) &= \sum_{\mathbf{x} \in \{S_n \cup S_m\}} \phi_n(\mathbf{x}) \log \frac{\phi_n(\mathbf{x})}{\phi_m(\mathbf{x})} \\ &= - \sum_{\mathbf{x} \in \{S_n \cup S_m\}} \phi_n(\mathbf{x}) \log \phi_m(\mathbf{x}) + \sum_{\mathbf{x} \in \{S_n \cup S_m\}} \phi_n(\mathbf{x}) \log \phi_n(\mathbf{x}) \end{aligned} \tag{8}$$

$$\|S_n - S_m\|_{KL} = D_{KL}(S_n \| S_m) + D_{KL}(S_m \| S_n) \tag{9}$$

The Parzen Window estimation of $\mathcal{P}^{\mathcal{S}}$ is defined as:

$$\Pr(\mathcal{S} = S) \simeq \frac{1}{|\mathcal{S}| h} \sum_{t,\theta} K \left(\frac{\|S - S_{t,\theta}\|_{KL}}{h} \right) \tag{10}$$

where $|\cdot|$ denotes the cardinality of $(\cdot)$ and K is some kernel function, for instance the Radial Basis Function:

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \tag{11}$$

The same method can be used to estimate $\phi_n(\mathbf{x})$ for a given subset S_n:

$$\phi_n(\mathbf{x}) = \frac{1}{|S_n|h} \sum_{\mathbf{x}_i \in S_n} K\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{h}\right) \tag{12}$$

Substituting 10 into 5 our probability distribution estimate becomes:

$$\varphi(\mathbf{x}, S) = \frac{1}{|\mathcal{S}|h} \sum_{t,\theta} K\left(\frac{\|\{S \cup \mathbf{x}\} - S_{t,\theta}\|_{KL}}{h}\right) \tag{13}$$

3.2 Multiple Channel Parzen Window

Extending the Parzen Window approach requires the realization that 8 requires that S_n and S_m both be defined over the same abstract space [1]. As mentioned earlier, in the Multiple Channel context, each channel is treated as an orthonormal basis of $\Omega^{\mathcal{S}}$. An obvious approach to defining a measure over $\Omega^{\mathcal{S}}$ for mutliple channels is to use the Euclidean norm of the Kullbeck Liebler divergence of each channel considered independently:

$$\begin{aligned} S_n^c &= [\mathbf{x} \mid \mathbf{x} \in \{S_n \cap X^c\}] \\ \|S_n - S_m\|_{KL}^2 &= \sqrt{\sum_c D_{KL}(S_n^c \| S_m^c)^2} \end{aligned} \tag{14}$$

This requires the following minor extension of 15 and 13:

$$\Pr(\mathcal{S} = S) \simeq \frac{1}{|\mathbf{S}|h} \sum_{t,\theta} K\left(\frac{\|S - S_{t,\theta}\|_{KL}^2}{h}\right) \tag{15}$$

$$\varphi(\mathbf{x}) = \frac{1}{|\mathbf{S}|h} \sum_{t,\theta} K\left(\frac{\|\{S \cup \mathbf{x}\} - S_{t,\theta}\|_{KL}^2}{h}\right) \tag{16}$$

[1] For instance if $X_n \in \mathbb{R}^2$ and $X_m \in \mathbb{R}^3$, it is impossible to calculate $\phi_n\big((x_m, t)\big)$ because the quantituy $\|x_m - x_n\|^2$ from using 12 is ambiguous.

4 Support Vector Estimation

The Parzen Window method is neither sparse nor computationally efficient, and as the number of observations grows, these deficiencies quickly become prohibitive. We now investigate the use of Support Vector Machines to generate $\varphi(\mathbf{x}, S)$.

4.1 Random Process Estimation

Support Vector Machines are usually used to estimate probability distributions by solving the related problem of estimating the cumulative distribution function of the random variable in question. This reduces the problem to one of estimating a non-linear mapping from observations to cumulative distribution values, which can be formulated as an optimization problem over a linear operator equation. Unfortunately, these methods depend on the ability to calculate an empirical distribution for each observation:

$$F_\ell(x) = \frac{1}{\ell} \sum_i \theta(x - x_i) \tag{17}$$

where $\theta(x)$ is the indicator function. To evaluate this function, the abstract space $\Omega^{\mathcal{S}}$ must be ordered. While we have described a distance metric over $\Omega^{\mathcal{S}}$, it is not clear what a meaningful ordering relation would be.

Rather than calculating the cumulative probability distribution of $\Omega^{\mathcal{S}}$, we begin with the assumption that the Parzen Window estimate of the probability distribution is accurate and attempt to minimize the square loss between the Support Vector estimate and the Parzen Window estimate. Because we are hoping to generate a sparse representation of the probability distribution, we add a regularizing term Ω which penalizes similar S. The Support Vector approach seeks a solution in the following form:

$$\begin{aligned} \varphi(\mathbf{x}, S) &= \sum_i \beta_i K(S_{\mathbf{x}}, S_i) \\ S_{\mathbf{x}} &= \{S \cup \mathbf{x}\} \end{aligned} \tag{18}$$

So we can express the Support Vector optimization problem as:

$$\begin{aligned} W(\beta) &= \sum_{\mathbf{x} \in X} \Big(\varphi_P(\mathbf{x}, S) - \varphi_{SV}(\mathbf{x}, S : \beta) \Big)^2 + \lambda \Omega(\beta, S) \\ &= \sum_{\mathbf{x} \in X} \Big(\varphi_{SV}(\mathbf{x}, S : \beta)^2 - 2\varphi_P(\mathbf{x}, S) \varphi_{SV}(\mathbf{x}, S : \beta) \Big) + \lambda \Omega(\beta, S) \\ &= \sum_{i,j} \beta_i \beta_j \sum_{\mathbf{x} \in X} K(S_{\mathbf{x}}, S_i) K(S_{\mathbf{x}}, S_j) - \sum_i \beta_i \sum_{\mathbf{x} \in X} \sum_j \frac{2}{|\mathcal{S}|} K(S_{\mathbf{x}}, S_i) K(S_{\mathbf{x}}, S_j) + \lambda \sum_i \beta_i \sum_{\mathbf{x} \in X} K(S_{\mathbf{x}}, S_i)^{-1} \\ &= \sum_{i,j} \beta_i \beta_j \sum_{\mathbf{x} \in X} K(S_{\mathbf{x}}, S_i) K(S_{\mathbf{x}}, S_j) + \sum_i \beta_i \sum_{\mathbf{x} \in X} \left(\lambda K(S_{\mathbf{x}}, S_i)^{-1} - \frac{2}{|\mathcal{S}|} \sum_j K(S_{\mathbf{x}}, S_i) K(S_{\mathbf{x}}, S_j) \right) \end{aligned}$$

$$\text{subject to} \quad \sum_i \beta_i = 1, \quad \beta_i \geq 0, \ i = 1, \ldots, |\mathcal{S}| \tag{19}$$

Notice the regularizer selected is defined as:

$$\Omega(\beta, S) = \sum_i \beta_i \sum_{\mathbf{x} \in X} K(S_{\mathbf{x}}, S_i)^{-1} \tag{20}$$

Equation 19 is a quadratic optimiztion problem defined as:

$$\begin{aligned} W(\beta) &= \frac{1}{2}\beta^T P \beta + q^T \beta \\ P_{i,j} &= \sum_{\mathbf{x} \in X} K(S_{\mathbf{x}}, S_i) K(S_{\mathbf{x}}, S_j) \\ q_i &= \sum_{\mathbf{x} \in X} \left(\lambda K(S_{\mathbf{x}}, S_i)^{-1} - \frac{2}{|\mathcal{S}|} \sum_j K(S_{\mathbf{x}}, S_i) K(S_{\mathbf{x}}, S_j) \right) \end{aligned}$$

$$P = \langle \mathbf{K}^T \cdot \mathbf{K} \rangle \tag{21}$$

$$q = \lambda \langle \mathbf{K}^T \cdot \mathbf{1}_{(|K|,1)} \rangle^{-1} - \frac{2}{|\mathcal{S}|} \langle \mathbf{K}^T \cdot \mathbf{K} \cdot \mathbf{1}_{(|K|,1)} \rangle \tag{22}$$

4.2 Random Process Kernel Definition

When developing the Parzen Window algorithm, we used kernel function 8 which is based on the Kullbeck Liebler divergence of a probability estimate 12 defined at the sets being evaluated. Our motivation in developing a Support Vector approach is to generate sparse representations of $\mathcal{P}^{\mathcal{S}}$, in part to reduce the computational demands of evaluating $\varphi(\mathbf{x}, S)$ for test data sets. Unfortunately, kernel function 8 requires probability estimates of both sets being compared - it would be helpful to develop a kernel function capable of evaluating a distance between a test set $S_{\hat{X}}$ and a training set S_i without first calculating an estimate of the probability distribution of $S_{\hat{X}}$.

Note that 19 is formulated in such a way that the first kernel argument is always $S_{\mathbf{x}}$, which allows us to define a kernel which takes a set of points as its first argument and a probability distribution as its second. The definition of 21 also allows us to define non-symmetric kernels without sacrificing convexity or monotonicity in the optimization objective function. We can therefore use any measure of the likelihood of observing a set of points given a probability distribution, for example the conditional likelihood, the shannon entropy, the jaynes entropy, the renyi entropy, and possibly the average self-information. We will use the Shannon Entropy, as it is equivalent the the Kullback Lieblier divergence in the context of Renyi entropy with $\alpha = 1$.

$$K_{SV}(X, \phi) = \sum_{x \in X} \phi(x) \log \phi(x) \tag{23}$$

Recalling that $S_{\mathbf{x}} = \{S \cup \mathbf{x}\}$, we observe that:

$$K(S_{\mathbf{x}}, S_i) = K(S, S_i) + K(\mathbf{x}, S_i) \tag{24}$$

In situations where probabilities are being calculated for a uniform set of points in $\Omega^{\mathcal{S}}$, this can considerably reduce the computational time needed to evaluate $\mathbf{K}$.

4.3 Random Vector Estimation

Evaluating 19 over a training set X produces a set $\mathcal{S}_{SV} \in \mathcal{S}$ referred to as Support Vectors. While 19 produces a sparse representation of $\mathcal{P}^{\mathcal{S}}$, further sparseness can be achieved by refining the definition of $\phi(\mathbf{x})$. In the context of computing the kernel matrices used in optimizing 19 the Parzen Window definition of $\phi(\mathbf{x})$ is computationally acceptable, as it results in a good approximation with a minimal amount of computation [2]. In the context of evaluating 18 over a testing set $\hat{X}$, the Parzen Window algorithm for $\phi(\mathbf{x})$ is sub-optimal due to the fact that it requires the full set of observations for each $S_i \in \mathcal{S}_{SV}$. In this context, a sparse algorithm for $\phi(\mathbf{x})$ would reduce both the data required to store $\mathcal{S}_{SV}$ and the computational resources required to evaluate **??**.
One method of Support Vector density estimation is to search for a solution in the following form:

$$p(x) = \sum_{i=1}^{\ell} \left(\alpha_i^1 \mathcal{K}_1(x_i, x) + ... + \alpha_i^{\kappa} \mathcal{K}_{\kappa}(x_i, x) \right) \tag{25}$$

by selecting values of β which minimize the square loss of the empirical cumulative probability distribution using some regularizer:

$$\min \left(\sum_{i=1}^{\ell} \left(y_i - \sum_{j=1}^{\ell} \sum_{n=1}^{\kappa} \alpha_j^n k_n(x_i, x_j) \right)^2 + \lambda \sum_{i=1}^{\ell} \sum_{n=1}^{\kappa} \frac{1}{\gamma_n} \alpha_i^n \right) \tag{26}$$

$$\text{subject to} \quad \sum_{i=1}^{\ell} \sum_{n=1}^{\kappa} \alpha_i^n = 1, \quad \alpha_i \geq 0 \tag{27}$$

$$\tag{28}$$

given a kernel function $k(x, x')$ and a cross-kernel $\mathcal{K}(x, x')$:

$$k(x, x') = \frac{1}{1 + e^{-\gamma(x - x')}} \tag{29}$$

$$\mathcal{K}(x, x') = -\frac{\gamma}{2 + e^{\gamma(x - x')} + e^{-\gamma(x - x')}} \tag{30}$$

[2] The Parzen Window estimate at a point requires distance computations for each observation and a summation. By contrast, the SV estimate requires the same number of distance computations as well as the solving of a quadratic optimization problem whose complexity increases exponentially with the number of observations.

4.4 Data Pre-Processing

5 Results

The architecture has been tested against several data sets. In all cases the system parameters are left unchanged to eliminate the possbility of optimizing the system performance to best match the known outcomes.

5.1 Eunite Competition Data

5.2 Santa Fe Data

5.3 CATS Benchmark Data

5.4 Results Summary

6 Further Research

6.1 Data Retention

6.2 Ensemble System

6.3 Sliding Prediction Offset

7 Conclusion

Eat it, bitches