Robust Timeseries Analysis in the Context of Multiple Asynchronous Input Channels

Ryan Michael
kerinin@gmail.com

March 7, 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 $t \in [0, 1)$:

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

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

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:

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

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:

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

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 $\phi_n(\mathbf{x}) \simeq Pr(X \mid S_n)$:

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

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

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:

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

This requires the following minor extension of 15 and 13:

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

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

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 Support Vector Optimization Formulation

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:

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

[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.

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:

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

So we can express the Support Vector optimization problem as:

$$\begin{aligned}W(\beta) &= \sum_{\mathbf{x}\in X} \Big(\varphi_P(\mathbf{x}, S) - \varphi_{SV}(\mathbf{x}, S:\beta)\Big)^2 + \lambda\Omega(\beta, S) \\ &= \sum_{\mathbf{x}\in X} \Big(\varphi_{SV}(\mathbf{x}, S:\beta)^2 - 2\varphi_P(\mathbf{x}, S)\varphi_{SV}(\mathbf{x}, S:\beta)\Big) + \lambda\Omega(\beta, S) \\ &= \sum_{i,j} \beta_i\beta_j \sum_{\mathbf{x}\in X} K(S_{\mathbf{x}}, S_i)K(S_{\mathbf{x}}, S_j) - \sum_i \beta_i \sum_j \sum_{\mathbf{x}\in X} \frac{2}{|\mathcal{S}|} K(S_{\mathbf{x}}, S_i)K(S_{\mathbf{x}}, S_j) + \lambda \sum_i \beta_i \sum_j K(S_i, S_j))^{-1} \\ &= \sum_{i,j} \beta_i\beta_j \sum_{\mathbf{x}\in X} K(S_{\mathbf{x}}, S_i)K(S_{\mathbf{x}}, S_j) - \sum_i \beta_i \sum_j \left(\lambda K(S_i, S_j)^{-1} + \frac{2}{|\mathcal{S}|} \sum_{\mathbf{x}\in X} K(S_{\mathbf{x}}, S_i)K(S_{\mathbf{x}}, S_j)\right) \\ &\text{subject to} \quad \sum_i \beta_i = 1, \quad \beta_i \geq 0,\ i = 1, \dots, |\mathcal{S}| \end{aligned} \tag{18}$$

Notice the regularizer selected is defined as:

$$\Omega(\beta, S) = \sum_i \beta_i \sum_j K(S_i, S_j)^{-1} \tag{19}$$

4.2 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, the 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 $S_{\hat{X}}$.

4.3 Single Channel Correlation

Given a set of estimations, we need a method to relate each estimate's relevance to observations in different time windows. We use the Shannon entropy of the set of observations in a given time window with respect to a given estimation as a measure of the estimate's relevance to the observed data. A set of data points which conform to the predicted values of an estimate will have lower entropy than a set of data points which diverge from the same estimate.

$$H_n(S_{t,\theta}) \mapsto \delta_{\phi_n,t,\theta} = \sum_{(x,t)\in S_{t,\theta}} -\phi_n(x,t)\log\phi_n(x,t) \tag{20}$$

$$\Delta_n = [\delta_{\phi_n,t,\theta} \mid \quad (t,\theta) \in \mathcal{T}]$$

The entropy for each estimate over different time windows defines a new random variable which we will refer to as H_n. We will compute a density estimate using ψ_C for H_n as we have with the input data previously:

$$H_n = (\Omega^H, \mathcal{F}^H, \mathcal{P}_n^H), \quad \Omega^H \in \mathbb{R}, \mathcal{F}^H = \mathcal{B}(\mathbb{R}) \tag{21}$$

$$\psi_C(\Delta_n, \alpha_C) \mapsto \varphi_n \tag{22}$$

$$\varphi_n(\delta) \simeq P\left(H_n = \delta\right) \tag{23}$$

4.4 Single Channel Pediction

The derivation algorithm produces predictions of X over a time window (t,θ) based on a set of observations which occur in that time window $S_{t,\theta}$. Rather than using the estimation algorithm ψ_E to predict these values, however, the derivation algorithm ψ_D predicts the probability distribution using bayesian modification of existing estimates. To accomplish this, we use the conditional probability of the entropy value.

$$P\left(X = (x,t) | H_n(S_{t,\theta}) = \delta\right) = \frac{P\left(H_n(S_{t,\theta}) = \delta | X = (x,t)\right) \cdot P\left(X = (x,t)\right)}{P\left(H_n(S_{t,\theta}) = \delta\right)}$$

Which for compactness we will write as:

$$P_{X|H_n}\left((x,t),\delta\right) = \frac{P_{H_n|X}\left(\delta,(x,t)\right) \cdot P_X\left((x,t)\right)}{P_{H_n}\left(\delta\right)} \tag{24}$$

We can easily determine one of these terms:

$$P_X\big((x,t)\big) = \prod_{0\le n<q} \phi_n(x,t) \tag{25}$$

Because the entropy 21 is calculated as a sum of the entropy of each observation, we can determine the conditional probability $P\left(H_n(S_{t,\theta}) = \delta | X = (x,t)\right)$ by subtracting the entropy of $H_n((x,t))$ from the value of δ, and then using our correlation estimate φ_n to predict the probability of $\delta - H_n((x,t))$. We include the expanded form of $P(H_n(S_{t,\theta}))$ to clarify the cancellation of terms:

$$\begin{aligned}
P_{H_n}\left(\delta\right) &= \frac{|y : H_n(y) = \delta|}{|y : H_n(y) \neq \delta|} \\
P_{H_n|X}\left(\delta, S_{t,\theta}\right) &= \frac{|y : H_n(y) = \delta + H_n\left((x,t)\right)|}{|y : H_n(y) \neq \delta|} \\
\frac{P_{H_n|X}\left(\delta, S_{t,\theta}\right)}{P_{H_n}(\delta)} &= \frac{|y : H_n(y) = \delta + H_n\left((x,t)\right)| \cdot |y : H_n(y) \neq \delta|}{|y : H_n(y) = \delta| \cdot |y : H_n(y) \neq \delta|} \\
&= \frac{P_{H_n}\left(\delta + H_n\left((x,t)\right)\right)}{P_{H_n}\left(\delta\right)}
\end{aligned}$$

Where $|\cdot|$ denotes the number of elements in the set $(\cdot)$. The derivation algorithm can easily be extended to multiple windows by using joint entropy probabilities. Because we do not know anything about the conditional relationships between estimates, we must assume they are i.i.d., and that their joint probability is equal to the sum of the marginal probabilities.

$$P(A|B \cap C) = \frac{P(B \cap C|A) \cdot P(A)}{P(B \cap C)}$$

$$\begin{aligned}
P_{X|H_n \cap H_m}\left((x,t), \delta_u, \delta_v\right) &= \frac{P_{H_n \cap H_m|X}\left(\delta_u, \delta_v, (x,t)\right) \cdot P_X\left((x,t)\right)}{P_{H_n \cap H_m}\left(\delta_u, \delta_v\right)} \\
P_{H_n \cap H_m|X}\left(\delta_u, \delta_v, (x,t)\right) &= \prod_{\substack{i=[n,m] \\ j=[u,v]}} P_{H_i}\left(\delta_j + H_i((x,t))\right) \\
&= \prod_{\substack{i=[n,m] \\ j=[i,j]}} \varphi_i\left(\delta_j + H_i((x,t))\right) \qquad (26) \\
P_{H_n \cap H_m}\left(\delta_u, \delta_v\right) &= \prod_{\substack{i=[n,m] \\ j=[u,v]}} P_{H_i}\left(\delta_j\right) \\
&= \prod_{\substack{i=[n,m] \\ j=[u,v]}} \varphi_i(\delta_j) \qquad (27)
\end{aligned}$$

Which gives us the following as our prediction algorithm:

$$\psi_P\left((x,t) : S_{t,\theta}, \Phi\right) \mapsto \mathbb{P} = \frac{\prod_{0 \le n < q} \varphi_n\big(H_n(S_{t,\theta})\big) \cdot \prod_{0 \le n < q} \phi_n(x,t)}{\prod_{0 \le n < q} \varphi_n\big(H_n(S_{t,\theta}) + H_n((x,t))\big)}$$

$$= \prod_{0 \le n < q} \phi_n(x,t) \cdot \frac{\varphi_n\Big(H_n(S_{t,\theta})\Big)}{\varphi_n\Big(H_n(S_{t,\theta}) + H_n\big((x,t)\big)\Big)} \tag{28}$$

4.5 Multiple Channel Estimation

Algorithms for determining probability density functions tend to scale exponentially with the dimensionality of the input data. For this reason it would be helpful if the algorithm could operate on independent channels of data and only calculate relationships between channels in cases where the channel's probability distribution is conditional on such relationships.

Extending the existing theory to this situation, we return to our original setting of the problem:

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

We will now refer to observations and estimates using superscript to denote their channel. We previously developed three generalized algorithms, ψ_E, ψ_C, ψ_D; we will now extend each one to handle multiple channels.

The estimation algorithm is unchanged in the context of multiple channels. Each channel generates independent estimates over independent time windows. The estimation process is intended to give each channel an understanding of its own behavior at a specific time window, and as such the generation of estimates does not rely on previous behaviors of the channel or the behavior of other channels. We therefore re-write the estimation algorithm to reflect the new notation:

$$S^A_{t,\theta} = \left[\left(x^A_i, \frac{t_i - t}{\theta}\right) \mid \quad x^A_i \in X^A,\ t \le t_i < t + \theta\right] \tag{29}$$

$$\psi_E(S^A_{t,\theta} : \alpha^A_E) \mapsto \phi^A_n \ \simeq \ \mathcal{P}^A(t,\theta) \tag{30}$$

$$\Phi^A = [\phi^A_0, ..., \phi^A_p]$$

$$\mathbf{\Phi} = [\Phi^0, ..., \Phi^c]$$

4.6 Multiple Channel Correlation

The correlation algorithm is where the most substantial changes must be made to accomodate multiple channels. The biggest difference from the single-channel approach is that each channel takes place in an independent abstract space. Recall that a critical component of the prediction algorithm is the evaluation of the entropy 21 of a set of observations given an estimate ϕ_n 31. If the probability density 31 operates over a different abstract space Ω than the observations $S_{t,\theta}$ are taken from, we cannot calculate the entropy. In other words, we cannot evaluate the entropy of an estimate from channel C^B using observations of channel C^A, so we must restrict our entropy definition to a single channel:

$$H_n^A(S_{t,\theta}^A) \mapsto \delta_{\phi_n,t,\theta}^A = \sum_{(x,t)\in S_{t,\theta}^A} -\phi_n^A(x,t)\log\phi_n^A(x,t) \tag{31}$$

In order to determine the conditional relations between channels, we begin with the observation that all channels share the time dimension t as part of their abstract space Ω^A. This allows us to specify a uniform time window (t,θ) for all channels $C^A \in \mathbf{C}$, and to then evaluate the entropy 32 of each estimate $\phi_n^A \in \Phi^A$ given the subset of each channel's observations $S_{t,\theta}^A$ for the time window. Given the time window (t,θ), we can treat these entropy measurements as a coherent set and interpret them as a multi-dimensional random vector in much the same way we treated different time windows in the single-channel case 22.

$$\Delta_{t,\theta} = \{A_{\phi_n} : \delta_{\phi_n,t,\theta}^A \mid \quad \phi_n \in \mathbf{\Phi},\ 0 \le A < c\}$$

Where $\Delta_{t,\theta}\{A_{\phi_n}\} = \delta_{\phi_n,t,\theta}^A$. Using the same algorithm ψ_C as we used previously, we can estimate the probability density φ of this random vector.

$$\begin{aligned} \mathbf{\Delta} &= [\Delta_{t,\theta} \mid \quad (t,\theta) \in \mathcal{T}] \\ \psi_C(\mathbf{\Delta},\alpha_C) &\mapsto \varphi \\ \varphi(\Delta_{t,\theta}) &\simeq \bigcap_{\substack{\phi_n\in\mathbf{\Phi} \\ 0\le A<c}} P(H_n^A = \Delta_{t,\theta}\{A_{\phi_n}\}) \end{aligned} \tag{32}$$

4.7 Multiple Channel Pediction

We begin by considering the situation in which we wish to make predictions on channel C^A using information from another, C^B. The derivation algorithm therefore produces predictions of X^A over a time window (t, θ) based on a set of observations which occur in that time window $S^B_{t,\theta}$. To do so we select estimates in both channels:

$$C^A \to [\phi^A_0, ..., \phi^A_p]$$
$$C^B \to [\phi^B_0, ..., \phi^B_q]$$

We calculate the entropy of each estimate in its own context:

$$[\delta^A_{\phi_0}, ..., \delta^A_{\phi_p}] = [H^A_0(S^A_{t,\theta}), ..., H^A_p(S^A_{t,\theta})]$$
$$[\delta^B_{\phi_0}, ..., \delta^B_{\phi_q}] = [H^B_0(S^B_{t,\theta}), ..., H^B_q(S^B_{t,\theta})]$$

Given an two estimates from the channels, we calculate the probability of each point in X^A based on the joint probability of the entropy values $P(\delta^A \cap \delta^B)$.

$$P^A_{X|H_n \cap H_m}\left((x,t), \delta^A, \delta^B\right) = \frac{P^A_{H_n \cap H_m|X}\left(\delta^A, \delta^B, (x,t)\right) \cdot P^A_X\left((x,t)\right)}{P^A_{H_n \cap H_m}\left(\delta^A, \delta^B\right)} \tag{33}$$

This time, we can easily determine two of these terms:

$$P^A_X\left((x,t)\right) = \prod_{0 \le i < p} \phi^A_i(x,t) \tag{34}$$

$$P^A_{H_n \cap H_m}\left(\delta^A, \delta^B\right) = \varphi\left(\{A_{\phi_n} : \delta^A, B_{\phi_m} : \delta^B\}\right) \tag{35}$$

Which leaves us with the process of calculating the effect of $X^A = (x,t)$ on the entropy of ϕ^A_n and ϕ^B_m. We adopt the same process used in the single-channel case, replacing δ^A with $\delta^A + H^A_n((x,t))$, and recalculating the joint probability. Since ϕ^B_m is in a different abstract space, we cannot make any useful statements about the dependence of its entropy given a point outside the abstract space being predicted. We can now determine the third of the three terms:

$$\begin{aligned} P^A_{H_n \cap H_m|X}\left(\delta^A, \delta^B, (x,t)\right) &= P^A_{H_n \cap H_m}\left(\delta^A + H^A_n((x,t)), \delta^B\right) \\ &= \varphi\left(\{A_{\phi_n} : \delta^A + H^A_n((x,t)), B_{\phi_m} : \delta^B\}\right) \end{aligned} \tag{36}$$

This result can easily be extended to an arbitrary number of channels and estimates:

$$P^A_\Delta\left(\Delta_{t,\theta}\right) = \varphi\left(\Delta_{t,\theta}\right) \tag{37}$$

$$P^A_{\Delta|X}\left(\Delta_{t,\theta}, (x,t)\right) = \varphi\left(\{A_{\phi_n} : \delta^A_{\phi_n} + H^A_n((x,t)), \Delta_{t,\theta} \wedge A_{\phi_n}\}\right) \tag{38}$$

Which gives us a the final prediction equation:

$$\begin{aligned} \psi(x,t) \mapsto \mathbb{P} &= P(X^A = (x,t)) \\ &= \prod_{0 \le n < p} \phi^A_n(x,t) \cdot \frac{\varphi\left(\{A_{\phi_n} : \delta^A_{\phi_n} + H^A_n((x,t)), \Delta_{t,\theta} \wedge A_{\phi_n}\}\right)}{\varphi(\Delta_{t,\theta})} \end{aligned} \tag{39}$$

5 Sample Implementation

Implementing the inference architecture requires the specification of the two density estimation functions, ψ_E and ψ_C. We will use the same algorithm for both; Vapnik's SVM for conditional density estimation. The SVM algorithm has been chosen due to its simplicity of parameters and determinacy.

5.1 Window Selection: One-Class SVM

`http://en.wikipedia.org/wiki/Semidefinite_embedding`

`http://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction`

`http://en.wikipedia.org/wiki/Feature_selection`

The first challenge is the selection of $\mathcal{T}$, the intervals used to define subsets $S_{t,\theta}$. Using the Support Vector approach, we search for a solution which represents the observed data as a linear combination of a Support Vectors drawn from the observed data. Rather than using the observations $X = [(x_0, t_0), ..., (x_\ell, t_\ell)]$, we use the subsets $\mathbf{S} = [S_{t,\theta} | t \in X, \theta \in \Theta]$, where Θ is a set of values which must be chosen. We select points from $\mathbf{S}$ to minimize the number of Support Vectors required to describe the observed data X to some level of accuracy.

We can treat each set $S \in \mathbf{S}$ as a probability measure over the abstract space Ω, and thus as a coding scheme for observations drawn from Ω. Our goal is to encode such observations using as few S as possible, so it is clear that maximizing the variation among the selected S is important.

The variation between any two sets S_i, S_j with corresponding probability measures p_i, p_j can be quantified using the Kullback Leibler divergence evaluated at each observation in the two sets:

$$D_{KL}(S_i || S_j) = -\sum_{x \in S_i \cup S_j} p_i(x) \log p_j(x) + \sum_{x \in S_i \cup S_j} p_j(x) \log p_i(x) \tag{40}$$

We use this function to define a kernel which assigns high values to sets which are similar and low values to sets which are different (the denominator is used to avoid a singuarity if $D_{KL} = 0$):

$$K(S_i, S_j) = \frac{1}{1 + D_{KL}(S_i || S_j) + D_{KL}(S_j || S_i)} \tag{41}$$

We define the optimization task as minimizing a weighted sum of the kernel matrix which corresponds to maximizing the divergence of the support vectors:

$$\min_\beta \sum_{i,j} \beta_i \beta_j K(S_i, S_j)$$

Using the One-Class SVM algorithm [?] to exclude a portion of outliers, we rewrite this objective function as

$$\min_\beta \sum_{i,j} \beta_i \beta_j K(S_i, S_j) - \sum_i \beta_i K(S_i, S_i) \tag{42}$$

$$\text{subject to} \quad 0 \leq \beta_i \leq \frac{1}{\nu \ell}, \quad \sum_i \beta_i = 1 \tag{43}$$

5.2 Estimate Generation: SVM-Density Algorithm

$$k(x,x') = \frac{1}{1+e^{-\gamma(x-x')}} \tag{44}$$

$$\mathcal{K}(x,x') = -\frac{\gamma}{2+e^{\gamma(x-x')}+e^{-\gamma(x-x')}} \tag{45}$$

$$\min \left(\sum_{i=1}^{\ell} \left(y_i - \sum_{j=1}^{\ell} \sum_{n=1}^{\kappa} \alpha_j^n k_n(x_i,x_j) \right)^2 + \lambda \sum_{i=1}^{\ell} \sum_{n=1}^{\kappa} \frac{1}{\gamma_n} \alpha_i^n \right) \tag{46}$$

$$\text{subject to} \quad \sum_{i=1}^{\ell} \sum_{n=1}^{\kappa} \alpha_i^n = 1, \quad \alpha_i \geq 0 \tag{47}$$

$$p(x) = \sum_{i=1}^{\ell} \left(\alpha_i^1 \mathcal{K}_1(x_i,x) + ... + \alpha_i^\kappa \mathcal{K}_\kappa(x_i,x) \right) \tag{48}$$

5.3 Correlation: SVM-Density Algorithm

5.4 Data Pre-Processing

6 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.

6.1 Eunite Competition Data

6.2 Santa Fe Data

6.3 CATS Benchmark Data

6.4 Results Summary

7 Further Research

The system as described thus far makes several assumptions for simplicity which would lead to unnecessary computational demands. We now spend some time discussion methods of reducing the computational demands of the system.

7.1 Data Retention

7.2 Estimation Optimization

The generation of estimates has been described as a process of selecting time windows at random from the full set of training data. Selecting windows at random is not necessary to satisfy the i.i.d. requirements of producing an accurate probability estimate, so long as the selection of time windows for which entropy is calculated is random. Let's consider the characteristics of a 'good' set of estimates for prediction.

The most obvious characteristic of a 'good' estimate is that it has a low average entropy, which is to say that it frequently captures the behavior of observed data. If an estimate is not applicable to the observed data, its influence on predictions will be marginal, and the computational time required to calculate its entropy and influence on a prediction will have been wasted.

Another characteristic of a 'good' set of estimates is that they have minimal redundancy. Again, if two estimates are essentially identical, their influence on prediction tasks will be identical, and the computational demands of evaluating their entropy and influence will be wasted.

Unfortunately, these two parameters will likely be mutually exclusive, so some method of balancing them is required.

7.3 Correlation

It is critical that correlation takes place using i.i.d. data in order to produce accurate probability estimates. We had previously assumed that each time window used to generate estimates would be used to correlate the estimates, however it is not necessary that the time windows used to determine the entropy values used to correlate estimates correspond to the time windows used to generate estimates. This allows us to select a number of time windows to use which balances the computational demands of density estimation with the need for accuracy.

7.4 Pediction

For prediction, it is not necessary to actually use each of the estimates in generating predictions, so long as we have accurate estimates of their individual probabilities and the joint probability of any two estimates. Since we must evaluate a number of joint probabilities equal to the square of the number of estimates being considered, selecting a useful subset of the defined estimates can provide substantial reductions in computational demands.

7.5 Ensemble System

7.6 Prediction with Heterogenous Time Windows

8 Conclusion

Eat it, bitches