Parzen-SVM: An Approach to Timeseries Analysis Using Support Vector Machines based on Parzen Windows

Ryan Michael
kerinin@gmail.com

March 12, 2009

Abstract

Contents

1 Introduction

1.1 Existing Work

1.1.1 Hidden Markov Model

1.1.2 Box-Jenkins

1.1.3 Spectral Analysis

1.1.4 Shrinking-ϵ SVM Regression

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.

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.

2 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 processes 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:

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

We define a set of time values and durations:

$$\mathcal{T} = [t_0, \dots, t_\ell]$$

$$\Theta = [\theta_0, \dots, \theta_z]$$

We assume that $\mathcal{P}$ can be approximated in a given time window using a shifted set of weighted distributions defined over some set of time intervals.

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

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

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

2.1 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:

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

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

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

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

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 $\mathcal{S}$:

$$S_{\hat{x}} = \Big[(x_i, t_i - t_{\hat{x}}) \mid \quad (x_i, t_i) \in \hat{X}\Big] \tag{4}$$

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 $\mathcal{S}$:

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

2.2 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:

$$S_{t,\theta} = \big[\, (x_i^n, t_i - t) \mid \quad x_i^n \in \mathbf{X},\ t \le t_i < t + \theta\big] \tag{6}$$

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:

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

3 Parzen Window Estimation

One method of evaluating 5 is by using the Parzen Window method. We choose the Parzen Window method because it allows us to estimate probabilities of unordered sets, provided they have an addition operation and a kernel function exists to provide a distance metric.

3.1 Single Channel Parzen Window

We will again begin by considering the single-channel case, then extend the resulting equations as necessary. 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 $\phi_n(\mathbf{x}) \simeq Pr(X = \mathbf{x} \mid S_n)$:

$$\begin{aligned} D_{KL}(S_n \| S_m) &= \sum_{\mathbf{x} \in \{S_n \cup S_m\}} \phi_n(\mathbf{x}) \log \frac{\phi_n(\mathbf{x})}{\phi_m(\mathbf{x})} \\ &= -\sum_{\mathbf{x} \in \{S_n \cup S_m\}} \phi_n(\mathbf{x}) \log \phi_m(\mathbf{x}) + \sum_{\mathbf{x} \in \{S_n \cup S_m\}} \phi_n(\mathbf{x}) \log \phi_n(\mathbf{x}) \qquad (8) \\ \|S_n - S_m\|_{KL} &= D_{KL}(S_n \| S_m) + D_{KL}(S_m \| S_n) \qquad (9) \end{aligned}$$

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

$$\Pr(\mathcal{S} = S) \simeq \sum_{i \in \mathcal{S}} \frac{1}{|\mathcal{S}|} K_\gamma(S, S_i) \tag{10}$$

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

$$K_\gamma(S_i, S_j) = \frac{1}{\gamma\sqrt{2\pi}} e^{-\frac{1}{\gamma}\|S_i - S_j\|^2_{KL}} \tag{11}$$

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

$$\phi_n(\mathbf{x}) = \sum_{\mathbf{x}_i \in S_n} \frac{1}{|S_n|} K_\gamma(\mathbf{x}, \mathbf{x}_i) \tag{12}$$

Substituting 10 into 5 our probability distribution estimate becomes:

$$\begin{gathered} \varphi_P(\mathbf{x}, S) = \sum_{i \in \mathcal{S}} \frac{1}{|\mathcal{S}|} K_\gamma(S_{\mathbf{x}}, S_i) \qquad (13) \\ S_{\mathbf{x}} = \{S \cup \mathbf{x}\} \end{gathered}$$

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:

$$S_n^c = [\mathbf{x} \mid \mathbf{x} \in \{S_n \cap X^c\}]$$

This requires the following minor extension of 11:

$$K_\gamma(S_i, S_j) = \prod_c \frac{1}{\gamma\sqrt{2\pi}} e^{-\frac{1}{\gamma}\|S_i^c - S_j^c\|_{KL}^2} \tag{14}$$

3.3 Sliding Parzen Windows

Observe that subsets do not exist in isolation; it is possible for the time windows defined for two subsets to overlap. This makes it possible for the estimates at at two such subsets to take advantage of this shared information. This can be accomplished by modifying 10 to include not only sets offset to the beginnig of the window being predicted, but at each start point defined as well:

$$S_{t,\theta}^{\tau} = [(x_i, 2t_i - t - \tau) \mid \quad (x_i^n, t_i) \in S_{t,\theta}]$$

$$\mathcal{S}^{\mathcal{T}} = [S_{t,\theta}^{\tau} | \quad \tau \in \mathcal{T},\ (t, \theta) \in (\mathcal{T}, \Theta)]$$

$$\Pr(\mathcal{S} = S) \simeq \sum_{i \in \mathcal{S}} \frac{1}{|\mathcal{S}^{\mathcal{T}}|} K_\gamma(S, S_i) \tag{15}$$

In the context of Parzen Windows, this defintion increases computational demands and provides little increase in predictive power, however we shall see that it provides an important starting point for improving system performance.

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 $\varphi(\mathbf{x}, S)$.

[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.

4.1 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:

$$F_\ell(x) = \frac{1}{\ell}\sum_i \theta(x - x_i) \tag{16}$$

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:

$$\begin{aligned} \varphi_{SV}(\mathbf{x}, S : \beta) &= \sum_{i \in \mathcal{S}} \beta_i K_\gamma(S_\mathbf{x}, S_i) \\ S_\mathbf{x} &= \{S \cup \mathbf{x}\} \end{aligned} \tag{17}$$

So we can express the Support Vector optimization problem as:

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

$$\text{subject to} \quad \sum_i \beta_i = 1, \quad \beta_i \geq 0, \; i = 1, \ldots, |\mathcal{S}| \tag{18}$$

Notice the regularizer selected is defined as:

$$\Omega(\beta, S) = \sum_{i \in \mathcal{S}} \beta_i \sum_{\mathbf{x} \in X} K_\gamma(S_\mathbf{x}, S_i)^{-1} \tag{19}$$

Equation 18 is a quadratic optimiztion problem defined as:

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

$$P = \langle \mathbf{K}^T \cdot \mathbf{K} \rangle \tag{20}$$

$$q = \lambda \langle \mathbf{K}^T \cdot \mathbf{1}_{(|K|,1)} \rangle^{-1} - \frac{1}{|\mathcal{S}|} \langle \mathbf{K}^T \cdot \mathbf{K} \cdot \mathbf{1}_{(|K|,1)} \rangle \tag{21}$$

4.2 Sliding Windows

Modifying the SV algorithm to use sliding time windows allows the system to take advantage of periodicity in the observed data and reduces redundancy. Adapting the earlier Parzen Window approach, we search for a solution in the following form:

$$\mathcal{S}^\mathcal{T} = [S^\tau_{t,\theta} | \quad \tau \in \mathcal{T}, \; (t, \theta) \in (\mathcal{T}, \Theta)]$$

$$\varphi_{SV}(\mathbf{x}, S : \beta) = \sum_{i \in \mathcal{S}^\mathcal{T}} \beta_i K_\gamma(S_\mathbf{x}, S_i) \tag{22}$$

The optimization problem must be modified as follows:

$$
\begin{aligned}
W(\beta) &= \sum_{\mathbf{x}\in X} \Big(\varphi_{SV}(\mathbf{x}, S:\beta)^2 - 2\varphi_{SV}(\mathbf{x}, S:\beta)\varphi_P(\mathbf{x}, S) \Big) + \lambda\Omega(\beta, S) \\
&= \sum_{\mathbf{x}\in X} \left(\sum_{i\in\mathcal{S}^{\mathcal{T}}} \beta_i K_\gamma(S_{\mathbf{x}}, S_i) \right) \cdot \left(\sum_{j\in\mathcal{S}^{\mathcal{T}}} \beta_j K_\gamma(S_{\mathbf{x}}, S_j) \right) \\
&\qquad\qquad - 2 \left(\sum_{i\in\mathcal{S}^{\mathcal{T}}} \beta_i K_\gamma(S_{\mathbf{x}}, S_i) \right) \cdot \left(\sum_{j\in\mathcal{S}^{\mathcal{T}}} \frac{1}{|\mathcal{S}^{\mathcal{T}}|} K_\gamma(S_{\mathbf{x}}, S_j) \right) + \lambda\Omega(\beta, S) \\
&= \sum_{i,j\in\mathcal{S}^{\mathcal{T}}} \beta_i\beta_j \sum_{\mathbf{x}\in X} K_\gamma(S_{\mathbf{x}}, S_i) K_\gamma(S_{\mathbf{x}}, S_j) \\
&\qquad\qquad + \sum_{i\in\mathcal{S}^{\mathcal{T}}} \beta_i \sum_{\mathbf{x}\in X} \left(\lambda K_\gamma(S_{\mathbf{x}}, S_i)^{-1} - \frac{2}{|\mathcal{S}^{\mathcal{T}}|} \sum_{j\in\mathcal{S}^{\mathcal{T}}} K_\gamma(S_{\mathbf{x}}, S_i) K_\gamma(S_{\mathbf{x}}, S_j) \right) \\
&\qquad\qquad \text{subject to} \quad \sum_i \beta_i = 1, \quad \beta_i \geq 0,\ i = 1, \ldots, |\mathcal{S}^{\mathcal{T}}| \qquad (23)
\end{aligned}
$$

This means that using sliding windows increases P by a factor of ℓ^2 and q by a factor or ℓ. Fortunately, most of the added kernel values will be 0 if we restrict $\theta_{\max}$ to some reasonable subset of the observed duration.

4.3 Parzen-SVM Decomposition

Due to the simplicity of the constraints in the optimization problem, it is possible to use the decomposition method of Osuna to reduce the memory requirements of the Parzen-SVM algorithm.

4.3.1 Sub-Problem Definition

The decomposition algorithm breaks X into two working sets B, N, and attempts to optimize B while keeping N fixed. This results in the following iterative optimization problem where $\boldsymbol{\beta}^k$ denotes the result of the previous iteration:

$$
\begin{aligned}
W(\boldsymbol{\beta}_B) &= \frac{1}{2} \begin{bmatrix} \boldsymbol{\beta}_B^T & (\boldsymbol{\beta}_N^k)^T \end{bmatrix} \begin{bmatrix} P_{BB} & P_{BN} \\ P_{NB} & P_{NN} \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta}_B \\ \boldsymbol{\beta}_N^k \end{bmatrix} - \begin{bmatrix} q_B^T & q_N^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta}_B \\ \boldsymbol{\beta}_N^k \end{bmatrix} \\
&= \frac{1}{2} \boldsymbol{\beta}_B^T P_{BB} \boldsymbol{\beta}_B - (-q_B + P_{BN}\boldsymbol{\beta}_N^k)^T \boldsymbol{\beta}_B \\
&= \frac{1}{2} \begin{bmatrix} \beta_i & \beta_j \end{bmatrix} \begin{bmatrix} P_{ii} & P_{ij} \\ P_{ij} & P_{jj} \end{bmatrix} \begin{bmatrix} \beta_i \\ \beta_j \end{bmatrix} - (-q_B + P_{BN}\boldsymbol{\beta}_N^k)^T \begin{bmatrix} \beta_i \\ \beta_j \end{bmatrix} \\
&\qquad \text{subject to} \quad 0 \leq \beta_i, \beta_j, \quad \beta_i + \beta_j = 1 - \mathbf{1}^T \boldsymbol{\beta}_N^k \qquad (24)
\end{aligned}
$$

4.3.2 Working Set Selection

Select

$$i \in \arg\max_t \left\{ -\nabla f(\boldsymbol{\beta}^k)_t \mid \quad t \in I(\boldsymbol{\beta}^k) \right\} \tag{25}$$

$$j \in \arg\min_t \left\{ -\frac{b_{it}^2}{a_{it}} \mid \quad t \in I(\boldsymbol{\beta}^k), \quad -\nabla f(\boldsymbol{\beta}^k)_t < -\nabla f(\boldsymbol{\beta}^k)_i \right\} \tag{26}$$

Where

$$I(\boldsymbol{\beta}) \equiv \{ t \mid \quad \beta_t < 1 \quad \text{or} \quad \beta_t > 0 \} \tag{27}$$

$$a_{it} = P_{ii} + P_{tt} - 2P_{it} \tag{28}$$

$$\bar{a}_{it} = \begin{cases} a_{it} & \text{if } a_{it} > 0 \\ \delta & \text{otherwise} \end{cases} \tag{29}$$

$$b_{it} = -\nabla f(\boldsymbol{\beta}^k)_i + \nabla f(\boldsymbol{\beta}^k)_t \tag{30}$$

$$\nabla f(\boldsymbol{\beta})_i \equiv P_i \boldsymbol{\beta} - q_i \tag{31}$$

4.3.3 Stopping Condition

$$\max_{i \in I(\boldsymbol{\alpha}^k)} -\nabla f(\boldsymbol{\alpha})_i + \min_{j \in I(\boldsymbol{\alpha}^k)} \nabla f(\boldsymbol{\alpha})_j \leq \epsilon \tag{32}$$

4.4 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 18 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 20 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 divergence between a set of points and a probability distribution. We consider the Renyi entropy with $\alpha = 1$ and select the Shannon Entropy as our kernel metric, as it is equivalent the the Kullback Lieblier divergence.

$$\|X - \phi\|_H = \sum_{x \in X} \phi(x) \log \phi(x) \tag{33}$$

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

$$\|S_{\mathbf{x}} - S_i\|_H = \|S - S_i\|_H + \|\mathbf{x} - S_i\|_H \tag{34}$$

We restate the kernel function as:

$$K_\gamma(X, S_i) = \frac{1}{\gamma\sqrt{2\pi}} e^{-\frac{1}{\gamma}\|X - S_i\|_H^2} \tag{35}$$

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 $\mathbf{K}$, as $\|\mathbf{x} - S_i\|_H$ can be computed once and then added to each $\|S - S_i\|_H, S \in \mathcal{S}$.

4.5 Random Vector Estimation

Evaluating 18 over a training set X produces a set $\mathcal{S}_{SV} \in \mathcal{S}$ referred to as Support Vectors. While 18 produces a sparse representation of $\mathcal{P}^{\mathcal{S}}$ by eliminating some S, further sparseness can be achieved by refining the definition of $\phi(\mathbf{x})$ to allow the elimination of some $\mathbf{x}$ in each $S \in \mathcal{S}_{SV}$. In the context of computing the kernel matrices used in optimizing 18 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 17 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 $S \in \mathcal{S}_{SV}$. 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 **??**.

We return now to the empirical cumulative distribution function method of Support Vector density estimation and search for a solution in the following form, which is able to use multiple non-symmetric kernel functions simultaneously:

$$\phi(x) = \sum_{i=1}^{\ell} \left(\beta_i^1 \mathcal{K}_1(x_i, x) + ... + \beta_i^\kappa \mathcal{K}_\kappa(x_i, x)\right) \tag{36}$$

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

[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.

$$\min \left(\sum_{i=1}^{\ell} \left(y_i - \sum_{j=1}^{\ell} \sum_{n=1}^{\kappa} \beta_j^n k_n(x_i, x_j) \right)^2 + \lambda \sum_{i=1}^{\ell} \sum_{n=1}^{\kappa} \frac{1}{\gamma_n} \beta_i^n \right) \tag{37}$$

$$\text{subject to} \quad \sum_{i=1}^{\ell} \sum_{n=1}^{\kappa} \beta_i^n = 1, \quad \beta_i \geq 0 \tag{38}$$

$$\tag{39}$$

given a kernel function $k(x, x')$ from the sigmoid family to approximate the cumulative probability distribution and it's derivative which we refer to as the cross-kernel $\mathcal{K}(x, x')$ which we use to construct an estimate of the probability distribution:

$$k(x, x') = \frac{1}{1 + e^{-\gamma(x-x')}} \tag{40}$$

$$\mathcal{K}(x, x') = -\frac{\gamma}{2 + e^{\gamma(x-x')} + e^{-\gamma(x-x')}} \tag{41}$$

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 Pre-Processing

6.2 Ensemble System

6.3 Weighted Influence

7 Conclusion

Eat it, bitches