Virtual Sensors: Using Data Mining Techniques to Efficiently Estimate Remote Sensing Spectra

Ashok N. Srivastava, *Member, IEEE,* Nikunj C. Oza, *Member, IEEE,* and Julienne Stroeve, *Member, IEEE*

***Abstract*— Various instruments are used to create images of the Earth and other objects in the universe in a diverse set of wavelength bands with the aim of understanding natural phenomenon. These instruments are sometimes built in a phased approach, with some measurement capabilities being added in later phases. In other cases, there may not be a planned increase in measurement capability, but technology may mature to the point that it offers new measurement capabilities that were not available before. In still other cases, detailed spectral measurements may be too costly to perform on a large sample. Thus, lower resolution instruments with lower associated cost may be used to take the majority of measurements. Higher resolution instruments, with a higher associated cost may be used to take only a small fraction of the measurements in a given area. Many applied science questions that are relevant to the remote sensing community need to be addressed by analyzing enormous amounts of data that were generated from instruments with disparate measurement capability. This paper addresses this problem by demonstrating methods to produce high accuracy estimates of spectra with an associated measure of uncertainty from data that is perhaps nonlinearly correlated with the spectra. We call this type of an estimator a *Virtual Sensor* because it predicts, with a measure of uncertainty, unmeasured spectral phenomenon.**

***Index Terms*— Data Mining, Neural Networks, Support Vector Machine, Kernel Methods, Remote Sensing.**

I. INTRODUCTION

THIS paper describes the development of data mining algorithms that learn to estimate unobserved spectra from remote sensing data. For purposes of the discussion presented here, we will model the data as matrices of time series (following the notation in [1]). The spatiotemporal random function $Z(\mathbf{u}, \lambda, t)$ is modelled as a finite number n of spatially correlated time series with the following representation:

$$\begin{aligned} Z(\mathbf{u}, \lambda, t) &= [Z_{\mathbf{u}}(\lambda, t)] \quad (1) \\ &= [Z_{u_1}(\lambda, t), Z_{u_2}(\lambda, t), ..., Z_{u_n}(\lambda, t)]^T \end{aligned}$$

In Equation 1, $\mathbf{u}$ represents the spatial coordinate, λ represents the vector of measured wavelength(s), and t represents time. The superscript ^{T} indicates the transpose operator. If multiple wavelengths are measured, then each Z_i is actually a matrix, and the function $Z(\mathbf{u}, \lambda, t)$ represents a data cube of size $(n \times \Lambda \times T)$, where these symbols represent the number of spatial locations, the total number of measured wavelengths, and the total number of time samples, respectively. In this notation, the spatial coordinate $\mathbf{u}$ represents the coordinates (or index) of a measurement at a particular location in the field of view and is not in any way related to the distributed nature of the data centers. Conceptually, the equation above describes a set of n $(\Lambda \times T)$ matrices. In the event that the spatial coordinate describes adjacent pixels, it is useful to think of Equation 1 as describing a time series of data cubes (spectral images) of size $n \times n \times \Lambda$.

Consider a situation where one is given a sensor $\mathcal{S}_1$ which takes k spectral measurements in wavelength bands $\mathbf{B}_1 = \{\lambda_1, \lambda_2, \ldots, \lambda_k\}$ at time t_1. Suppose that we have another sensor $\mathcal{S}_2$ which has a set of spectral measurements taken at time t_2, $\mathbf{B}_2 = \{\lambda_1, \lambda_2, \ldots, \lambda_k, \lambda_{k+1}, \lambda_{k+2}, \ldots, \lambda_{k+l}\}$ that partially overlaps the spectral features contained in $\mathbf{B}_1$ in terms of power in the spectral bands. Thus, we would have $\mathbf{B} = \mathbf{B}_1 \setminus \mathbf{B}_2 = \{\lambda_{k+1}, \lambda_{k+2}, \ldots, \lambda_{k+l}\}$ representing the common spectral measurements. Note that these measurements are common only in their power In this situation, we investigate the problem of building an estimator $\Gamma(Z(\mathbf{B}))$ that best approximates the joint distribution $P(Z(\mathbf{B})|Z(\mathbf{B}_1))$. Thus, we would have:

$$\Gamma(Z(\mathbf{B})) \approx P(Z(\mathbf{B})|Z(\mathbf{B}_1)) \quad (2)$$

The value of building an estimator for P is clear particularly in situations where one sensor has been in operation for a much longer period of time than another. The first sensor may have fewer spectral channels in which measurements are taken compared to the newer sensor. However, it may be of scientific value to be able to estimate what the spectral measurements in wavelengths $\mathbf{B}$ would have been had the first sensor had the measurement capability.

The joint distribution given by $P(Z(\mathbf{B})|Z(\mathbf{B}_1))$ above contains all necessary information to recover the underlying structure captured by the sensor $\mathcal{S}_2$. If perfect reconstruction of this joint distribution were possible, we would no longer need sensor $\mathcal{S}_2$ because all relevant information could be generated from the smaller subset of spectral measurements $\mathbf{B}_1$ and the estimator Γ. Of course, such estimation is often extremely difficult because there is not sufficient information in the bands $\mathbf{B}_1$ to perfectly reconstruct the distribution. Also, in many cases, the joint distribution cannot be modelled properly using parametric representations of the probability distribution since that may require a significant amount of domain knowledge and may be a function of the ground cover, climate, sun position, time of year, and numerous other factors.

In this paper, we describe methods to estimate the first moments of this distribution. Some methods that we use allow

Manuscript received March 15, 2004; revised November 18, 2002. This work was supported by the NASA Intelligent Systems Intelligent Data Understanding Program.

A. N. Srivastava and N. C. Oza are at the NASA Ames Research Center. J. Stroeve is with the National Snow and Ice Data Center

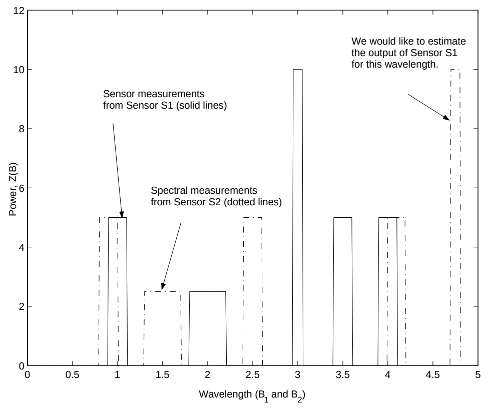

Fig. 1. This figure helps illustrate the need for a Virtual Sensor. We have spectral measurements from two sensors $\mathcal{S}_1$ and $\mathcal{S}_2$, (solid and dotted lines, respectively). We wish to estimate the output of sensor $\mathcal{S}_1$ for a wavelength where there is no actual measurement from the sensor. Note that some sensor measurements overlap perfectly, as in the case of wavelength $= 3$, and in other cases, such as wavelength = 1, there is some overlap in the measurements.

us to model the second moment of the distribution as well:

$$\begin{aligned}\mu(\mathbf{B}) &= \int \Gamma(\mathbf{B})\mathbf{B}d\mathbf{B} \\ \sigma^2(\mathbf{B}) &= \int [\Gamma(\mathbf{B}) - \mu(\mathbf{B})]^2\mathbf{B}d\mathbf{B}\end{aligned}$$

We use the function Γ in the above computations as an estimate of the (unknown) joint distribution P. Several computational problems as well as problems due to the underlying physical measurement process arise when we attempt to estimate Γ. We begin by describing some of the problems that may arise due to the physical aspects of the two measurement devices and then discuss computational considerations.

Figure 1 gives a schematic view of the problem. The solid and dotted lines correspond to sensors $\mathcal{S}_1$ and $\mathcal{S}_2$ respectively. A Virtual Sensor can be built when there are some overlapping sensor measurements as depicted in the figure. Notice that if there are no overlapping sensor measurements, we are unable to build an estimator. In real-world problems, some measurements may overlap perfectly, while others have a partial overlap. Generally speaking the measurements from Sensor $\mathcal{S}_1$ are not available at all wavelength locations.

In the event that the spectral measurements are perfectly overlapping for all k wavelength bands and the measurements for sensor $\mathcal{S}_1$ are not available at the remaining $\mathbf{B}$ bands, the estimation process is more straightforward. When partial overlap occurs between two sensors for a given wavelength, calculations need to be performed to estimate the amount of power that would have been measured in the overlapping bands. This can be done using interpolation methods.

Situations such as this often arise in practice. For example consider the relationship between the AVHRR (Advanced Very High Resolution Radiometer) and the MODIS (Moderate Resolution Imaging Spectroradiometer) instruments. Specifically, we show how to create a so-called Virtual Sensor to model MODIS Channel 6 as a function of other MODIS channels that are also available in AVHRR. This way, the created model can be used to construct the virtual AVHRR channel 6 as a function of the other channels available in AVHRR. Data mining methods are tested and their results examined for this task. New data mining algorithms are developed based on these results. Because of the large amounts of AVHRR and MODIS data available, the algorithm will focus development on producing high-quality results efficiently and quickly with principled estimates of uncertainty. Clearly, the construction of a Virtual Sensor has two key components. The first is constructing the model that generates the Virtual Sensor data given the known data. This requires training data—data for which there are true sensor values corresponding to the values of the Virtual Sensor. In this example, we would use MODIS images to generate a model that predicts MODIS channel 6 as a function of the other MODIS channels that are also available in AVHRR. Only channels common to MODIS and AVHRR can be used because of the second component of virtual sensor construction: generating the virtual sensor values. The learned model has to be used to generate the AVHRR virtual channel 6 as a function of the other AVHRR channels.

Some preliminary studies were made to check the feasibility of the Virtual Sensor using some MODIS and AVHRR images acquired over the Greenland ice sheet. In particular, supervised learning methods (e.g., neural networks) are capable of using MODIS data to construct a model that can predict MODIS channel 6 as a function of other MODIS channels. This model can then take an AVHRR image as input and can construct the virtual channel 6.

II. Virtual Sensors for Cryosphere Analysis

Intensification of global warming in recent decades has caused a rise of interest in year-to-year and decadal-scale climate variability in the Polar Regions. This is because these regions are believed to be one of the most sensitive and vulnerable regions to climatic changes. The enhanced vulnerability of the Polar Regions is believed to result from several positive feedbacks, including the temperature-albedo-melt feedback and the cloud-radiation feedback. Recent observations of record regional anomalies in ice extent, thinning of the margins of the Greenland ice sheet, and reduction in the northern hemispheric snow cover, may reflect the effect of these feedbacks. Remote sensing products now provide spatially and temporally continuous and consistent information on several polar geophysical variables over nearly three decades. This period is sufficiently long enough to permit evaluation of how several cryospheric variables change in phase with each other and with the atmosphere and can help to improve our understanding of the processes in the coupled land-ice-ocean-atmosphere climate system. Cloud detection particularly over snow- and ice-covered surfaces is difficult using sensors such as AVHRR. This is because of the lack of spectral contrast between clouds and snow in the channels flown on the earlier AVHRR/2 sensors. Snow and clouds are both highly reflective in the visible wavelengths and often show little contrast in the thermal infrared.

The AVHRR Polar Pathfinder Product (APP) consists of twice daily gridded (at 1.25 and 5km spatial resolution) surface albedo and temperature from 1981 to 2000. A cloud mask accompanies this product but has been found to be inadequate, particularly over the ice sheets (Stroeve [2001]). The 1.6

micron channel on the MODIS instrument as well as the AVHRR/3 sensor can significantly improve the ability to detect clouds over snow and ice. Therefore, by developing a virtual sensor to model the MODIS 1.6 micron channel as a function of the AVHRR/2 channels, we can improve the cloud mask in the APP product, and subsequently improve the retrievals of surface temperature and albedo in the product. In doing so we will be able to improve the accuracy in documenting seasonal and inter-annual variations in snow, ice sheet and sea ice conditions since 1981.

III. Creating a Virtual Sensor

In this section we outline the procedure for creating a Virtual Sensor. At minimum, we assume that for sensor $\mathcal{S}_1$ we have measurements $Z_1(\mathbf{B}_1)$ from one image, and for another sensor $\mathcal{S}_2$ we assume that we have another image $Z_2(\mathbf{B}_2)$. The procedure for creating a Virtual Sensor is as follows, assuming that we need to build a predictor for channel $\mathbf{b}_{k+1}$:

1) Divide the data set $Z_2(\mathbf{B}_2)$ into a training set and a test set.

2) Find parameters θ that minimize the squared error (or another suitable metric) $[E[\Gamma(Z_2(\mathbf{B}_1),\theta)] - Z_2(\mathbf{b}_{k+1})]^2$.

3) Apply Γ to the data from sensor $\mathcal{S}_1$ to generate an estimate of $E[\Gamma(Z_1(\mathbf{b}_{k+1}),\theta)]$. This is the step where the estimation of the unknown spectral contribution occurs.

4) Evaluate the results based on science based metrics and other information known about the image.

The procedure described above is standard in the data mining literature. From the remote sensing perspective, it is interesting to see the potentially systematic differences between the performance of the estimator on data from sensors 1 and 2.

Note that this procedure will only work if sufficient information exists to predict $Z(\mathbf{B})$ given data $Z(\mathbf{B}_1)$. One simple procedure for determining this is to look at the linear correlation between the spectra. The top panel of Figure 2 shows the inter-channel linear correlation for the first seven channels of MODIS data. Larger squares indicate stronger linear correlation. Red squares indicate negative correlation and green squares indicate positive correlation. The lower panel in this figure shows the results of computing the mutual information between the pairs of channels. The mutual information between two random variables is given by:

$$I(x,y) = \sum_{i=1}^{N}\sum_{j=1}^{M} P(x_i,y_j)\log\frac{P(x_i,y_j)}{P(x_i)P(y_j)} \tag{3}$$

This method gives a nonlinear measure of the relationship between the channels. Again, the larger the square, the greater the degree of relationship. In the case described in this paper, we will be building models in order to predict Channel 6. Notice that Channel 6 has small linear correlations with the other channels but moderate mutual information.

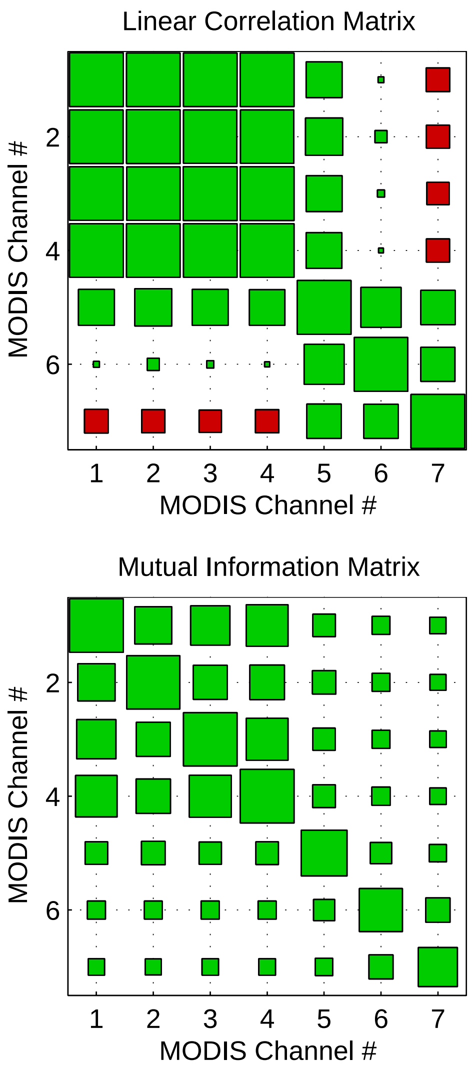

Fig. 2. The upper panel of this figure shows the linear correlation between the first seven channels of the MODIS instrument for one point in time. The size of the square indicates the degree of linear correlation. Green color indicates a positive correlation, and red color indicates a negative correlation. Notice that Channel 6, which is a channel that we will try to emulate using a Virtual Sensor, has a relatively weak correlation with the other channels. The lower panel indicates the mutual information between the same MODIS channels. Notice that for this nonlinear measure of information, channel 6 has more relationship with the other channels, thus giving hope that a nonlinear model could be built to predict channel 6.

The next section describes three estimation methods that we have used to build a Virtual Sensor: a feed-forward neural network (also called a multilayer perceptron, (MLP)), a Support Vector Machine (SVM), and an SVM with a Mixture Density Mercer Kernel.

IV. Standard Data Mining Methods

There are many machine learning methods that have been used in many different types of problems. We give a brief review of the methods that we use in this paper.

We first describe multilayer perceptrons, a type of neural network [Bishop, 1995]. The central idea of neural networks is to construct linear combinations of the inputs as derived features, and then model the target as a nonlinear function of these derived features. Neural networks are often depicted as

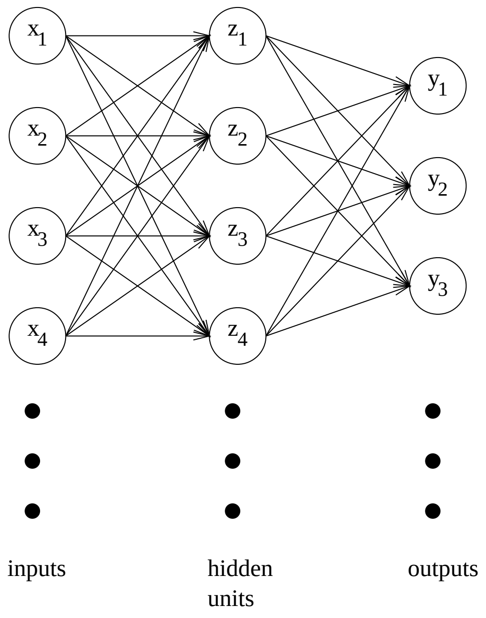

Fig. 3. An example of a MultiLayer Perceptron (MLP).

a directed graph consisting of nodes and arcs. An example is shown in Figure 3. Each column of nodes is a layer. The leftmost layer is the input layer. The inputs of an example to be classified are entered into the input layer. The second layer is the hidden layer and the third layer is the output layer. Information flows from the input layer to the hidden layer and then to the output layer via a set of arcs (depicted in figure 3 as arrows). Note that the nodes within a layer are not directly connected. In our example, every node in one layer is connected to every node in the next layer, but this is not required in general. Also, a neural network can have more or less than one hidden layer and can have any number of nodes in each hidden layer.

Each non-input node, its incoming arcs, and its single outgoing arc constitute a neuron, which is the basic computational element of a neural network. Each incoming arc multiplies the value coming from its origin node by the weight assigned to that arc and sends the result to the destination node. The destination node adds the values presented to it by all the incoming arcs, transforms it with a nonlinear activation function (to be described later), and then sends the result along the outgoing arc. For example, the return value of a hidden node z_j in our example neural network is

$$z_j = g\left(\sum_{i=1}^{|A|} w_{i,j}^{(1)} x_i\right),$$

where $|A|$ is the number of input units, $w_{i,j}^{(k)}$ is the weight on the arc in the kth layer of arcs that goes from unit i in the kth layer of nodes to unit j in the next layer (so $w_{i,j}^{(1)}$ is the weight on the arc that goes from input unit i to hidden unit j) and g is a nonlinear activation function. A commonly used activation function is the sigmoid function:

$$g(a) \equiv \frac{1}{1+exp(-a)}.$$

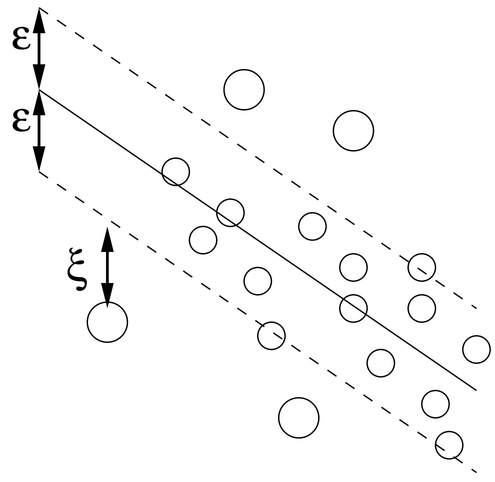

Fig. 4. Support Vector Machine for regression. The solid line is the line fitted to the points (represented as circles). The dashed lines are a distance ϵ from the fitted line. The points within the dashed line are considered to have zero error by an ϵ-insensitive loss function.

The return value of an output node y_j is

$$y_j = g\left(\sum_{i=1}^{Z} w_{i,j}^{(2)} z_i\right)$$

where Z is the number of hidden units. The outputs are clearly nonlinear functions of the inputs.

Neural networks are trained to fit data by a process that is essentially nonlinear regression. Given each entry in the training dataset, the network's current prediction is calculated. The difference between the true function value and the prediction is the error. The derivative of this error with respect to each weight in the network is calculated and the weights are adjusted accordingly to reduce the error.

A. Support Vector Machines

Support Vector Machines for classification and regression are described in detail in [**?**], but here we briefly describe Support Vector Regression (SVR), which we use in this paper. In real-world problems, traditional linear regression cannot be expected to fit a set of points perfectly (i.e., with zero error). For this reason, nonlinear regression is often used with the hope that a more powerful nonlinear model will achieve a better fit than a linear model. However, this power often comes with two drawbacks. One is that the space of parameters of a nonlinear model (such as the multilayer perceptrons discussed above) often have many local optima that are not globally optimal. Nonlinear regression algorithms such as backpropagation for MLPs often find these local optima, which can result in a model that does not predict well on unseen data. The second drawback is that nonlinear model fitting is often overly sensitive to the locations of the training points, so that they overfit the training points and do not perform well on new data.

SVR addresses these problems in three ways. The first way is to use an ϵ-insensitive loss function. If y is the true response and $f(\mathbf{x})$ is the predicted response for the input $\mathbf{x}$, then the loss function is

$$|y - f(\mathbf{x})|_\epsilon = max\{0, |y - f(\mathbf{x})| - \epsilon\}$$

That is, if the error between the true response and the predicted response is less than some small ϵ, then the error on that point is considered to be zero. For example, in figure 4, the solid line, which is the fitted line, is within ϵ of all the points between the two dashed lines; therefore, the error is considered to be zero for those points. If ϵ is set to the level of the typical noise that one can expect in the response variable, then support vector regression is less likely to expend effort fitting the noise in the training data at the expense of generalization performance, i.e., it is less likely to overfit. In particular, to estimate the linear regression

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$$

one solves the optimization problem of minimizing

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} |y_i - f(\mathbf{x}_i)|_\epsilon,$$

where C is a user-determined constant that determines the tradeoff between the closeness of the fit (second term) and the level of regularization (first term).

The second way support vector regression addresses the above problems is to allow some error beyond ϵ for each training point but minimize the total such error over all the points. In figure 4, ξ is the additional error for one particular point. For m training points, define ξ_i for $i \in \{1, 2, \ldots, m\}$ to be the slack variables that represent the additional allowable error if $f(\mathbf{x}_i) - y_i > \epsilon$ (i.e., $\xi_i = 0$ otherwise) and ξ_i^* to be the additional error if $y_i - f(\mathbf{x}_i) > \epsilon$. In that case, the optimization problem is the following:

$$\begin{aligned} \text{minimize } \frac{1}{2}\|\mathbf{w}\|^2 \quad &+ \quad C\sum_{i=1}^{m}(\xi_i + \xi_i^*) \\ \text{subject to} \quad & \\ f(\mathbf{x}_i) - y_i \quad &\leq \quad \epsilon + \xi_i \\ y_i - f(\mathbf{x}_i) \quad &\leq \quad \epsilon + \xi_i^* \\ \xi_i \quad &\geq \quad 0 \\ \xi_i^* \quad &\geq \quad 0 \\ \text{for all } i \in 1, 2, \ldots, m. & \end{aligned}$$

Note that the above optimization problem will involve minimizing the sum of the slack variables. Also, any points for which the error is already less than ϵ will end up with zero for their corresponding slack variables. Because this is a convex optimization problem, there is a unique globally optimal solution.

The third way that SVR addresses the above problems is to map the data from the original data space into a much higher (possible infinite) dimensional *feature space* and calculate the support vector machine in that space. The idea is that the linear model in the feature space may correspond to a complicated nonlinear model in the original data space. Clearly, one needs a practical way to deal with data that is mapped to such a high-dimensional space, which intuitively seems impossible. However, one is able to do this using the kernel trick. By introducing Lagrange multipliers and obtaining the dual of the previous optimization problem (see [?] for the details), one obtains the following:

$$\begin{aligned} \text{maximize}_{\alpha,\alpha^* \in \mathcal{R}} & -\epsilon\sum_{i=1}^{m}(\alpha_i^* + \alpha_i) + \sum_{i=1}^{m}(\alpha_i^* - \alpha_i)y_i \\ & -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)\mathbf{x}_i \cdot \mathbf{x}_j \\ & \text{subject to } 0 \leq \alpha_i, \alpha_i^* \leq C \text{ for all } i \in \{1, 2, \ldots, m\} \\ & \text{and } \sum_{i=1}^{m}(\alpha_i - \alpha_i^*) = 0. \end{aligned}$$

The resulting regression estimate is of the form

$$f(\mathbf{x}) = \sum_{i=1}^{m}(\alpha_i^* - \alpha_i)\mathbf{x}_i \cdot \mathbf{x}_j + b.$$

Note that the inputs only appear as dot products in the above solution. Therefore, one can map the inputs into a very high or even infinite dimensional space H using a function $\Phi : \mathcal{R}^d \rightarrow H$ and the dot product $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ will still be a scalar. Of course, Φ would be too difficult to work with because of the high dimensionality of H. However, there exist *kernel functions* $K(x_i, x_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ such that K is practical to work with even though the Φ induced by that K is not. For example, the Gaussian kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|}{2\sigma^2}}$$

gives rise to a Φ that is infinite-dimensional. However, we do not need to deal with Φ or even know what it is because the Φ's only appear as dot products, which can be replaced by K. Therefore, the new regression estimate after mapping the inputs from the data space to the feature space is

$$f(\mathbf{x}) = \sum_{i=1}^{m}(\alpha_i^* - \alpha_i)K(\mathbf{x}_i, \mathbf{x}_j) + b.$$

In summary, the Support Vector Machine allows us to fit a nonlinear model to data without the local optima problem that other procedures suffer from.

The kernel function can be viewed as a measure of similarity between two data points. For example, with the Gaussian kernel, the value increases as the distance between the pair of points decreases. There is significant current research attempting to determine which kernel functions are most appropriate for different types of problems. One such novel kernel function is the Mixture Density Mercel Kernel (MDMK) which is discussed in the next section.

V. Mixture Density Mercer Kernels

The idea of using probabilistic kernels was discussed by Haussler in 1999 [2] where he observes that if $K(\mathbf{x}_i,\mathbf{x}_j) \geq 0$ $\forall$ $(\mathbf{x}_i,\mathbf{x}_j) \in \mathcal{X} \times \mathcal{X}$, and $\sum_{\mathbf{x}_i}\sum_{\mathbf{x}_j} K(\mathbf{x}_i,\mathbf{x}_j) = 1$ then K is a probability distribution and is called a P-Kernel. He further observed that the Gibbs kernel $K(\mathbf{x}_i,\mathbf{x}_j) = P(\mathbf{x}_i)P(\mathbf{x}_j)$ is also an admissible kernel function.

Our idea is to use an ensemble of probabilistic mixture models as a similarity measure. Two data points will have a larger similarity if multiple models agree that they should be placed in the same cluster or mode of the distribution. Those points where there is disagreement will be given intermediate similarity measures. The shapes of the underlying mixture distributions can significantly affect the similarity measurement of the two points. Experimental results uphold this intuition and show that in regions where there is “no question” about the membership of two points, the Mixture Density Kernel behaves identically to a standard mixture model. However, in regions of the input space where there is disagreement about the membership of two points, the behavior may be quite different than the standard model. Since each mixture density model in the ensemble can be encoded with domain knowledge by constructing informative priors, the Mixture Density Mercer Kernel (MDMK) will also encode domain knowledge. The MDMK is defined as follows:

$$\begin{aligned} K(\mathbf{x}_i,\mathbf{x}_j) &= \Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j) \\ &= \frac{1}{Z(\mathbf{x}_i,\mathbf{x}_j)}\sum_{m=1}^{M}\sum_{c_m=1}^{C_m} P_m(c_m|\mathbf{x}_i)P_m(c_m|\mathbf{x}_j) \end{aligned}$$

The feature space is thus defined explicitly as follows:

$$\begin{aligned} \Phi(\mathbf{x}_i) \propto\ & [P_1(c=1|\mathbf{x}_i), P_1(c=2|\mathbf{x}_i), \ldots, \\ & P_1(c=C|\mathbf{x}_i), P_2(c=1|\mathbf{x}_i), \ldots, P_M(c=C|\mathbf{x}_i)] \end{aligned}$$

The first sum in the defining equation above sweeps through the M models in the ensemble, where each mixture model is a Maximum A Posteriori estimator of the underlying density trained by sampling (with replacement) the original data. We will discuss how to design these estimators in the next section. C_m defines the number of mixtures in the mth ensemble, and c_m is the cluster (or mode) label assigned by the model. The quantity $Z(\mathbf{x}_i,\mathbf{x}_j)$ is a normalization such that $K(\mathbf{x}_i,\mathbf{x}_i) = 1$ for all i. The fact that the Mixture Density Kernel is a valid kernel function arises directly from the definition.

The Mixture Density Kernel function can be interpreted as follows. Suppose that we have a hard classification strategy, where each data point is assigned to the most likely posterior class distribution. In this case the kernel function counts the the number of times the M mixtures agree that two points should be placed in the same cluster mode. In soft classification, two data points are given an intermediate level of similarity (between 0 and 1) which will be less than or equal to the case where all models agree on their membership, in which case the entry would be unity. Further interpretation of the kernel function is possible by applying Bayes rule to the defining equation of the Mixture Density Kernel. Thus, we have:

TABLE I
An Example of a Table

One	Two
Three	Four

$$\begin{aligned} K(\mathbf{x}_i,\mathbf{x}_j) &= \frac{1}{Z(\mathbf{x}_i,\mathbf{x}_j)}\sum_{m=1}^{M}\sum_{c_m=1}^{C_m} \frac{P_m(\mathbf{x}_i|c_m)P_m(c_m)}{P_m(\mathbf{x}_i)} \times \\ &\quad \frac{P_m(\mathbf{x}_j|c_m)P_m(c_m)}{P_m(\mathbf{x}_j)} \\ &= \frac{1}{Z(\mathbf{x}_i,\mathbf{x}_j)}\sum_{m=1}^{M}\sum_{c_m=1}^{C_m} \frac{P_m(\mathbf{x}_i,\mathbf{x}_j|c_m)P_m^2(c_m)}{P_m(\mathbf{x}_i,\mathbf{x}_j)} \end{aligned}$$

The second step above is valid under the assumption that the two data points are independent and identically distributed. This equation shows that the Mixture Density Kernel measures the ratio of the probability that two points arise from the same mode, compared with the unconditional joint distribution. If we simplify this equation further by assuming that the class distributions are uniform, the kernel tells us on average (across ensembles) the amount of information gained by knowing that two points are drawn from the same mode in a mixture density.

VI. Conclusion

The conclusion goes here.

Appendix I
Proof of the First Zonklar Equation

Appendix one text goes here.

Appendix II

Appendix two text goes here.

Acknowledgment

The authors would like to thank...

References

[1] P. C. Kyriakidis and A. G. Journel, “Geostatistical space-time models: A review,” *Mathematical Geology*, vol. 31, no. 6, pp. 651–684, 1999.

[2] D. Haussler, “Convolution kernels on discrete structures,” University of California Santa Cruz, Tech. Rep., 1999.

PLACE
PHOTO
HERE

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.