Determining Intent using Hard/Soft Data and Gaussian Process Classifiers

Steven Reece and Stephen Roberts
Robotics Research Group
Dept. Engineering Science
Oxford University, UK
Email: {reece, sjrob}@robots.ox.ac.uk

David Nicholson and Chris Lloyd
BAE Systems
Advanced Technology Centre
PO Box 5, Filton, Bristol, UK
Email: {david.nicholson2, chris.m.lloyd}@baesystems.com

***Abstract*—Modern applications of data fusion are rarely starved of data but they look more challenging because the data can be diverse (hard and soft), uncertain and ambiguous, and often swamped by irrelevant detail. This paper presents a mathematical framework for dealing with these issues. It manages diverse data by representing it in the common format of a kernel matrix. Uncertainty is managed by interpreting the kernels as the covariance matrices of Bayesian Gaussian Processes. Finally, irrelevant detail is managed by automatically detecting the relevant (or, conversely, the irrelevant) components within a multi-source dataset. The framework is illustrated by applying it to a synthetic vehicle-borne Improvised Explosive Device intent recognition scenario. The results provide a proof-of-principle and encourage future work to develop practical implementations of algorithms in support of sense making and intelligence analysis.**

Keywords: Detection, Classification, Learning, Data Fusion, Gaussian Processes

1. Introduction

With the ongoing proliferation of sensor types and sensing modalities in civilian and military settings, information fusion must be able to deal with large volumes of heterogeneous data. In particular, the 'hard' data associated with physical sensor devices has become increasingly augmented by 'soft' data associated with human sources. Often, these data types complement one another because hard sources make objective measurements of target states (e.g. identity, position) whereas soft sources make subjective judgements about target states (e.g. intent). Thus a combination of sources is likely to improve target state estimation in terms of overall completeness and accuracy [1].

The complex target of interest in this paper is the 'pattern of life' associated with a counter-insurgency (COIN) scenario involving vehicle movements between specific buildings in a fictitious town. Based on surveillance reports from hard and soft sources stationed in the town, intelligence analysts would like to classify this pattern as a signature of either benign or hostile intent. For example, routine delivery of goods to a shopping mall in the former case or vehicle-borne delivery of an Improvised Explosive Device (IED) to the mall in the latter. The focus of this paper is the formulation and proof-of-principle testing of a mathematical framework for representing and fusing heterogeneous data sources to provide inference in support of intelligence analysts.

A kernel-based mathematical framework is described here for representing disparate hard/soft data as well as performing inference with this data. This framework can be used to design regressor and classifier algorithms, with a specific focus in this paper on binary classification problems though this is easily extendable. The kernel-based approach is able to quantify the different forms of uncertainty inherent in the data. Further, as the method is Bayesian it provides a principled way of quantifying uncertainty in the inferences. Finally, the approach is able to determine the relevant information sources within a potentially large array of data streams. Consequently, the intelligence analyst is able to assign limited sensing resources to the most informative data streams.

The paper is organised as follows. Section 2 outlines related work in the algorithmic and application areas of interest. The next two sections describe the core components of our framework – kernel representations, Gaussian Processes, and Automatic Relevance Determination. Section 5 describes an illustrative COIN scenario that was used to test the framework, with experimental results and analysis reported in Section 6. Finally, Section 7 contains the main conclusions and future research challenges.

2. Related Work

Hard-soft fusion is an emerging topic of both theoretical and applied importance within the information fusion community. This has been recognised by the recent publication of a textbook on the subject [1], a special session at Fusion 2010 [2], a multi-million dollar Multidisciplinary University Research Initiative on network-based hard-soft information fusion funded by the US Army Research Office, and the creation of a large hard-soft dataset for promoting algorithm development and analysis [3].

A mathematical framework for heterogeneous data fusion can be approached from either a parametric or a non-parametric modelling perspective. The parametric approach specifies fixed parametric models of the state space based on domain knowledge, whereas in the non-parametric approach these models are determined from data.

A common parametric approach to tackle information fu-

sion problems is to represent them as probabilistic graphical models, such as Kalman Filters, Hidden Markov Models, or richer Bayesian Networks [4]. These models encode domain knowledge in the structure of the graph and employ Bayesian methods to fuse conditional probability data at the nodes. In practice, parametric models may be very difficult to construct and implement due to limited domain knowledge. This is of particular concern in the complex and asymmetric COIN domain.

Random Finite Set theory (RFS) is a promising candidate theory for the fusion of disparate information [5]. RSF transforms all types of data, whether qualitative statements or quantitative values, into a common quantitative representation, the random finite set. Fuzzy logic, rough set theory, possibility theory and probability theory all have a corresponding formulation within RFS theory. Consequently, RFS can encode the disparate forms of uncertainty inherent in the data and inference can be performed using all the data available. However, the RFS requires parametric models to transform data into random finite sets. These can be learned from a sufficient supply of training data.

A non-parametric approach is investigated here, motivated by the requirement for flexibility in the absence of deep domain models and recognising the trend for ever increasing volumes of heterogeneous data with which to potentially learn models in the application domains of interest. For the COIN domain, the specific approach being sought must also satisfy the following requirements:

- Representation of heterogeneous (hard and soft) data in a common mathematical format;
- Automatic detection and exclusion of irrelevant data;
- Maintenance and evaluation of uncertainty and risk throughout.

While non-parametric methods have a long history of application in data mining and machine learning, there is currently a gap for a framework that addresses all these requirements. The purpose of this paper is therefore to present such a framework and provide some initial proof-of-principle testing.

3. Kernels for Heterogeneous Data

Given multiple heterogeneous data inputs relating to a scenario, the first requirement is to represent them in a common mathematical format. This section describes a kernel representation of data which represents data independent of its type in the common format of a kernel matrix.

A scenario may be described succinctly by a heterogeneous *feature vector*. A feature vector can contain hard and soft data, sequential events, relationships between events or virtually any information relating to the scenario. For example, the feature vector $\mathbb{V} = \langle$'Jim', 1604, 'large bag', 'soon afterwards'$\rangle$ describes the scenario 'Jim arrived at 1604 with a large bag. He left soon afterwards'. Let ϕ_i denote a function which extracts the i$^{\text{th}}$ feature from the feature vector:

$$\phi_i(\mathbb{V}) \triangleq [\mathbb{V}]_i \ .$$

It is assumed that each feature vector contains the same sequence of data types. That is, $\phi_i(\mathbb{V})$ has the same type for all $\mathbb{V}$. This allows individual features to be compared between feature vectors using distance measures tailored to specific types of data, whether it is nominal, ordinal, hard or soft. It is also assumed that each feature vector is complete and does not contain missing values. Relaxing these assumptions is a subject for future research.

Entire feature vectors may be compared by using kernels which transform individual features of different types into a common space. Each kernel, K, takes as input two values, $x_1 = \phi_i(\mathbb{V}_1)$ and $x_2 = \phi_i(\mathbb{V}_2)$, corresponding to the same feature in different feature vectors and returns a value indicating how close the two features actually are. There are various kernels for representing heterogeneous data types. Common examples include:

The Squared Exponential Kernel:

$$K_{SE}(x_1, x_2) = A \exp\left(-\frac{(x_1 - x_2)^2}{L^2}\right)$$

for any $x_1, x_2 \in \mathbb{R}$. The Squared Exponential kernel provides a smooth proximity measure over the continuous quantitative inputs x. Here, $L \geq 0$ and $A \geq 0$ are hyperparameters and are called the *input scale* and the *output scale* respectively.

The Nominal Kernel:

$$K_N(x_1, x_2) = A \exp\left(-\frac{I(x_1, x_2)}{L}\right)$$

where the indicator function $I(x_1, x_2) = 1$ if x_1 and x_2 are the same and zero, otherwise. The inputs, x, can take any nominal values including tags. Again, the input scale L specifies the degree of similarity of the inputs and is learned from the training data.

The Rank Kernel:

$$K_R(x_1, x_2) = A \exp\left(-\frac{M(x_1, x_2)}{L}\right)$$

for any ranked inputs where $R(x)$ indicates the rank of x within the rank order. The rank distance $M(x_1, x_2) = |R(x_2) - R(x_1)|$ is the absolute difference between the input rankings.

Kernels may also be placed over probability distributions. For example, let $P_l(h \mid C)$ and $P_m(h \mid C)$ be the probabilities of an object having height, h, when an individual or a population uses the words 'largish' or 'medium', respectively, in some context C. We may use the Hellinger distance (or many others [6]) to determine the correlation between the outputs of two objects, x_1 and x_2, whose heights are qualitatively labelled $Q(x_1) \in \{l, m\}$ and $Q(x_2) \in \{l, m\}$, respectively:

$$K_P(x_1, x_2) = A \exp\left(-\frac{H(x_1, x_2)}{L}\right)$$

and $H(x_1, x_2) = \frac{1}{2} \int dh (\sqrt{P_{Q(x_1)}(h \mid C)} - \sqrt{P_{Q(x_2)}(h \mid C)})^2$.

Kernels can be combined in a number of ways, for example, by addition:

$$K_{new}(x_1, x_2) = K_1(x_1, x_2) + K_2(x_1, x_2) + \ldots + K_n(x_1, x_2).$$

or by multiplication:

$$K_{new}(x_1, x_2) = K_1(x_1, x_2) \times K_2(x_1, x_2) \times \ldots \times K_n(x_1, x_2).$$

In general, a kernel sum is appropriate for combining disjunctive inputs for which different outputs are correlated if *any* element of the input vectors are close. A kernel product is more appropriate when outputs are correlated only if the entire input vectors are close.

Kernel combination may be used to place kernels over entire feature vectors. The feature kernel, $K_{\mathbb{V}}(\mathbb{V}_1, \mathbb{V}_2)$, measures the closeness of two feature vectors, $\mathbb{V}_1$ and $\mathbb{V}_2$. Firstly, choose an appropriate kernel, K_i, for each feature, $\phi_i(\mathbb{V})$. Then individual kernels are combined, for example, using the product rule into a single feature vector kernel:

$$K_{\mathbb{V}}(\mathbb{V}_1, \mathbb{V}_2) = \prod_i K_i(\phi_i(\mathbb{V}_1), \phi_i(\mathbb{V}_2)) \ .$$

The product of kernels is more appropriate for representing the COIN scenario investigated later in Section 5 as we classify behaviour as benign or hostile based on *all* relevant information. That is, a test feature vector will share the same class label as a training sample only when all the relevant features in the test scenario are 'close' to the corresponding features in the training sample.

4. Gaussian Processes

The second requirement is to identify a mathematical approach to heterogeneous data fusion using kernels which offers a principled approach to the evaluation of uncertainty and risk.

The *Gaussian Process* (GP), Support Vector Machines and Kernel Density Estimators are all examples of kernel based methods. GPs were chosen as the basis for our mathematical framework because they provide a principled Bayesian approach to uncertainty and also impose an intuitive interpretation on the kernel, K. In GPs the kernel is often called the *covariance function* as $K(x_1, x_2)$ is the *covariance* between the output values at x_1 and x_2.

4.1. Gaussian Process Regression

A GP is often regarded as a 'Gaussian distribution over functions' [7]. It can be thought of as the generalisation of a Gaussian distribution over a finite vector space to a function space of infinite dimension. Just as a Gaussian is fully specified by its mean and covariance matrix, a Gaussian process is fully described by its mean and covariance function K. Although these functions can be infinitely dimensional, GPs are used to infer, or predict, function values at a finite set of *test points* using the observed data. The training data $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ is drawn from a noisy process:

$$y_i = f(x_i) + \epsilon_i \tag{1}$$

where ϵ_i is independent identically distributed Gaussian noise with variance σ^2. For convenience both inputs and outputs are aggregated into $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_n\}$ respectively. The GP estimates the value of the function f at arbitrary test points $X_* = \{x_{*1}, \ldots, x_{*m}\}$. The basic GP regression equations are given in [7]:

$$\bar{f}_* = m(X_*) + K(X_*, X)[K(X, X) + \sigma^2 I]^{-1} \times (Y - m(X)) \ , \tag{2}$$

$$\text{Cov}(f_*) = K(X_*, X_*) - K(X_*, X) \times [K(X, X) + \sigma^2 I]^{-1} K(X_*, X)^T \tag{3}$$

where I is the identity matrix, $m(\cdot)$ is the prior function mean[1], $\bar{f}_*$ is the posterior function mean and $\text{Cov}(f_*)$ is the posterior covariance. The matrix $K(X, X)$ denotes the joint prior distribution covariance of the function at inputs X. This covariance matrix has elements:

$$K(x_i, x_j) = \text{Cov}(f(x_i), f(x_j)) \ .$$

The matrix $K(X_*, X)$, obtained from the kernel K, is the covariance between the function at the prediction points, X_*, and the training inputs, X. Many covariance functions have been designed to capture various properties of the modelled phenomenon including smoothness, periodicity and stationarity. They are constructed to guarantee that any matrix obtained from them is a covariance matrix (i.e. positive semi-definite) irrespective of the choice of inputs X. Kernels can be linearly or multiplicatively combined to form new kernels as described in Section 3.

Each kernel has a set of hyperparameters, θ, which include the input and output scales (and also σ). We write $K(X, X, \theta)$ when referring to specific hyperparameter values. The most likely values for these hyperparameters, along with σ, can be estimated from the marginal likelihood:

$$\theta = \text{argmax}_{\theta'} Pr(Y \mid X, \theta')$$

where $Pr(Y \mid X, \theta') = \mathbb{N}(Y; 0, K(X, X, \theta'))$.

4.2. The Binary Gaussian Process Classifier

The key problem addressed in this paper is how to classify feature vectors into one of two classes denoted $y = 0$ or $y = 1$, indicating a hostile or benign event, for example. The GP regressor considered so far can be extended to classification problems [7]. As identical scenarios are re-encountered they may exhibit behaviours associated with each class occasionally. For example, a small van driven by a nervous driver may only occasionally be hostile, but not necessarily all of the time. To accommodate a distribution over class labels for each scenario we model the probability of class membership, $Pr(y = 1 \mid x)$ for each scenario x. For binary classification $Pr(y = 0 \mid x) = 1 - Pr(y = 1 \mid x)$.

The scenarios, x, are expressed as feature vectors and form the inputs of our GP classifier. The outputs are the probability

[1] Often a zero prior function is chosen: $m(x) = 0$ for all x.

distributions $Pr(y = 1 \mid x)$. Since the output is a probability function then the output space must be confined to the range $[0, 1]$. Following [7] we choose to map from the Gaussian process latent regressor function, f, onto $y \in [0, 1]$ via the sigmoid function, $g(f(x))$:

$$g(f(x)) = \frac{1}{1 + \exp(-sf(x))}$$

where $s > 0$ is the sigmoid *sensitivity*. Also, following [7], we define the latent function so that it has a direct probabilistic interpretation:

$$Pr(y = 1 \mid x) = g(f(x)) \ .$$

The sigmoid serves a further purpose, it allows near step-wise changes in probability across class boundaries whilst requiring only smoothly varying functions in the latent space. Thus, we can use simple off-the-shelf covariance functions, K, such as the squared-exponential, without the need to define change-points or declare kernel non-stationarities at the class boundaries. The sigmoid function approximately maps a normal distributed latent function into a Beta distributed class probability, $Pr(y = 1 \mid x)$, for each scenario, x, since:

$$\mathbb{B}(g(f(x)))\frac{dg(f(x))}{df(x)} \approx \mathbb{N}(f(x); m_x, P_x) \tag{4}$$

where $Pr(y = 1 \mid x) = g(f(x))$ and m_x and P_x are the latent function mean and variance for some normal distribution at x.

We will now show that updating a Beta distribution in probability space is approximately equivalent to updating a normal distribution in latent space. Let the training samples, D, be binary samples drawn for various scenarios. Let's now focus on those samples of scenario x alone and the impact that these data have on the posterior distributions, $Pr(y = 1 \mid x)$ and $Pr(y = 1 \mid x')$, for different scenarios, $x' \neq x$. These samples comprise N_x occurrences of scenario x where n_x of these samples are assigned class label $y(x) = 1$ and $N_x - n_x$ are assigned class label $y(x) = 0$. The posterior Beta distribution over $Pr(y = 1 \mid x)$ can be inferred using Bayes rule:

$$\begin{aligned}\mathbb{B}(Pr(y = 1 \mid x) \mid n_x, N_x, \cdot) \propto Pr[n_x, N_x \mid Pr(y = 1 \mid x)] \\ \times \mathbb{B}(Pr(y = 1 \mid x) \mid \cdot) \ .\end{aligned} \tag{5}$$

We approximate (5) via normal distributions. Since the 'class' draws are conditionally independent then $Pr[n_x, N_x \mid Pr(y = 1 \mid x)] \propto Pr_{\text{Binomial}}(n_x; N_x, Pr(y = 1 \mid x))$. The expected number of occurrences of $y(x) = 1$ is $N_x Pr(y = 1 \mid x)$ with variance $N_x Pr(y = 1 \mid x)(1 - Pr(y = 1 \mid x))$. We can approximate the Binomial likelihood by a normal likelihood by moment matching:

$$\begin{aligned}&Pr_{\text{Binomial}}(n_x; N_x, Pr(y = 1 \mid x)) \\ &\approx \mathbb{N}(n_x; N_x Pr(y = 1 \mid x), N_x Pr(y = 1 \mid x)(1 - Pr(y = 1 \mid x))) \ .\end{aligned}$$

This approximation is reasonably accurate for all N_x but improves for larger N_x. Scaling n_x by N_x, we have:

$$Pr_{\text{Binomial}}(n_x; N_x, Pr(y = 1 \mid x)) \approx \mathbb{N}(z_x; \mu_x, V_x) \tag{6}$$

where:

$$z_x = \frac{n_x}{N_x} \ , \tag{7}$$

$$\mu_x = Pr(y = 1 \mid x) \ , \tag{8}$$

$$V_x = \frac{Pr(y = 1 \mid x)(1 - Pr(y = 1 \mid x))}{N_x} \ . \tag{9}$$

The class probability, $Pr(y = 1 \mid x)$, which is necessary to calculate the variance, V_x, can be estimated from the training data. We use the expectation of the Beta distribution over $Pr(y = 1 \mid x)$ given the training data, D, to determine an approximate value for $Pr(y = 1 \mid x)$:

$$\hat{Pr}(y = 1 \mid x) = \frac{n_x + 1}{N_x + 2} \ . \tag{10}$$

Substituting (4) and (6) into (5) and then dividing throughout by $\frac{dg(f(x))}{df(x)}$ from (4), we have:

$$\mathbb{N}(f(x); m_x, P_x) \propto \mathbb{N}(z_x; g(f(x)), V_x)\mathbb{N}(f(x); m'_x, P'_x) \tag{11}$$

where m'_x and P'_x are the prior Gaussian mean and variance, respectively, and m_x and P_x are the posterior mean and variance. Equation (11) demonstrates that updating a Beta distribution in probability space is approximately equivalent to updating a normal distribution in latent space.

The posterior class probability for scenarios other than x are impacted by observations $z(x)$ via GP interpolation in the latent space. It remains to show how to infer the multivariate latent function, $f(X^*)$, for the inputs X^*, given data $z(X)$. Invoking Bayes rule:

$$\begin{aligned}Pr[f(X^*) \mid z(X)] \propto \int Pr[f(X^*) \mid f(X)] \ Pr[z(X) \mid f(X)] \\ \times Pr[f(X)] \ df(X) \ .\end{aligned}$$

Since each datum, z_x, is conditionally independent (and using (6)):

$$\begin{aligned}Pr(z(X) \mid f(X)) &= \prod_{x \in X} Pr(z_x \mid f(x)) \\ &\approx \prod_{x \in X} \mathbb{N}(z_x; g(f(x)), V_x)\end{aligned}$$

where V_x is defined in (9). Thus, the posterior Beta distribution over $Pr(y = 1 \mid x^*)$ is updated using the appropriate Binomial likelihood function when direct observations of the class of x^* are made. However, the posterior Beta distribution is also moderated by observations of closely related scenarios through Gaussian process interpolation in the latent space.

As $f(X)$ is a Gaussian process but the sigmoid function, g, is non-linear, the posterior $Pr(f(X^*) \mid z(X))$ can be inferred using the Extended Kalman Filter (EKF). Within the EKF implementation the latent function values of the test points, $X^* = \{x^*_1, \ldots, x^*_L\}$, are stacked to form the L *state vector*,

$f(X^*)$. The prior mean over $f(X^*)$ is chosen to be zero as this induces the appropriate prior mean of 0.5 for the class probabilities $Pr(y = 1 \mid x^*)$ for all $x^* \in X^*$. The EKF initial state covariance, K, is exactly the latent function Gaussian process covariance kernel. This is the feature vector product kernel described in Section 3.

The sample class counts, $z_x = n_x/N_x$, for M feature vectors $x \in X$ in the training set are also stacked, $z(X)$. Consequently, the EKF equations for the mean, $\hat{f}(X^*)$, and covariance, Cov$(f)(X^*)$, for the Gaussian, $Pr(f(X^*) \mid D)$ at the test points are:

$$\hat{f}(X^*) = W\ G^T\ (z(X) - 0.5)\ , \tag{12}$$

$$\text{Cov}(f)(X^*) = K(X^*, X^*) - W\ G\ K(X, X^*) \tag{13}$$

where the $L \times M$ matrix, W is the Kalman gain $W = K(X^*, X)\ G^T\ (G\ K(X, X)\ G^T + Q)$, the $M \times M$ matrix G is the diagonal sigmoid Jacobian matrix with diagonal elements:

$$G(x, x) = s\ g(f(x))\ (1 - g(f(x))) \tag{14}$$

and the $M \times M$ matrix Q is the diagonal count variance matrix with elements:

$$Q(x, x) = V_x\ . \tag{15}$$

We approximate the sigmoid Jacobian (14) at the expected probability $g(f(x)) = \hat{Pr}(y = 1 \mid x)$ defined in (10).

The class probability for a test point, x^*, is obtained by marginalising the latent function at x^*:

$$\hat{Pr}(y = 1 \mid x^*, D) = \int g(f(x^*)) Pr(f(x^*) \mid D)\ df\ .$$

Following [8], we approximate the posterior class mean as follows (where $f(x^*) \sim \mathbb{N}(\hat{f}(x^*), P(x^*))$):

$$\hat{Pr}(y = 1 \mid x^*) = g(\kappa(x^*)\hat{f}(x^*)) \tag{16}$$

where:

$$\kappa(x^*) = \left(1 + \frac{\pi P(x^*)}{8}\right)^{-1/2}\ .$$

An alternative Expectation-Propagation approach to latent function inference in classification problems is presented in [7].

4.3. Automatic Relevance Determination (ARD)

Determining the relevant features is crucial to successful classification. If we include irrelevant features in the feature vector then class uncertainty will increase. This arises when different feature values for irrelevant features increase the distance between, otherwise close, feature vectors. However, excluding relevant features also causes class uncertainty to increase. This arises because subtle distinctions between classes will be lost if relevant data is excluded. Fortunately, identifying the relevant features is straightforward within the GP classifier algorithm.

Following [9] when a feature is irrelevant its input scale is very large. For very large input scales the covariance will be independent of that input, effectively removing it from the feature vector. Our classifier performs ARD after the most likely feature input scales have been estimated. The marginal likelihood for the most likely input scales is λ. Each feature is considered in turn and the likelihood, λ_f, is re-calculated when the input scale for that feature is artificially set to a large value. The ratio $\rho_f = \lambda/\lambda_f$ is a measure of the relevance of feature f as it indicates the classifier performance gain when feature f is used by the classifier. A feature with $\rho_f < 1$ is detrimental to the classifier as its inclusion in the feature vector leads to poorer performance. Consequently, the relevant features are those for which $\rho_f > 1$. [2]

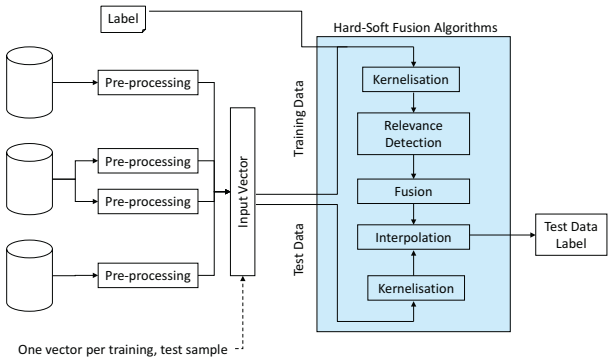

Figure 1. Information architecture relating the heterogeneous input data on the left to the inference of a class label for a test data sample on the right via a process of kernelisation, kernel combination, relevance detection, and GP classification.

The overall architecture for implementing relevance detection and the other parts of our heterogeneous data fusion framework is show in Figure 1. The input data, on the left, consists of multi-source data for a number of training and test scenarios. This is pre-processed with feature extractors appropriate for each source, such as image processing to extract texture features or document processing to extract word count features. All of the feature data is then represented by kernels. The kernels for the labelled training data are combined and relevant features are determined. The label for the test data is then inferred from the training data by the Gaussian Process classifier.

5. Illustrative COIN Scenario

The COIN scenario used to illustrate the application of our mathematical framework is a vehicle-borne IED threat detection scenario, designed originally to test parametric modelling techniques [10]. The scenario, displayed in Figure 2 involves a number of building facilities and a number of vehicles in transit between those facilities carrying out various functions: transport, collection, and delivery. If the intent of the

[2] We note that, in the rare case when two important features are completely correlated, $\rho_f = 1$ for both. Both will be excluded using our ARD mechanism. A more robust, yet computationally less efficient approach would be to employ an iterative ARD scheme which only removes one feature with $\rho_f = 1$ at each iteration. Subsequent iterations would perform ARD over the remaining potentially relevant features.

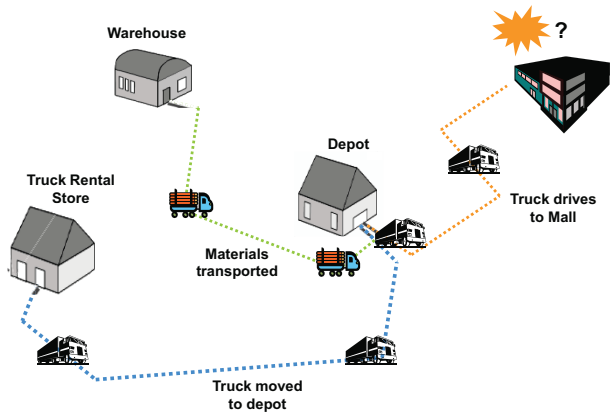

Figure 2. COIN scenario example [10]. It is required to infer whether the observed pattern of vehicle movement between the buildings is indicative or hostile or benign intent.

participants of these collective events is hostile, they culminate in the delivery of an IED to a shopping mall. However, it is more likely that the collective events are benign and result in the delivery of garden and electrical supplies to the mall for reasons of routine business and commerce.

It is assumed that this urban 'hot spot' has been under observation by the authorities for some time and certain reports have been logged. Moreover, each of these reports is labelled according to a forensic analysis which determined whether it formed part of a hostile or benign scenario. Consequently, there is a set of labelled exemplars for each scenario class with which to train the GP classifier to recognise new scenarios.

The simulated dataset for this scenario consisted of a mix of hard (quantitative) and soft (qualitative) report types. In total there are ten observables associated with the scenario, seven hard and three soft. The hard data elements refer to time stamps associated with the various vehicle arrival, departure and wait times during the scenario. Each hard type is modelled using a Squared Exponential kernel (see Section 3). The soft elements refer to the sizes of the vehicles {very small, small, medium, large, very large}, which are modelled using the Rank kernel, and the emotional state of the driver of the vehicle departing the depot {calm, nervous}, which is modelled by the Nominal kernel.

6. Experimental Results

The first experiment tests the claimed ability of our framework to automatically determine relevant sources of data. For this purpose an experimental dataset was generated by creating a number of hostile and non-hostile scenario exemplars in the ratio 1:4. The scenarios were created in such a way that only four of the ten observable features (unknown to the algorithms) distinguish whether they are hostile or non-hostile. For example, a medium-sized vehicle is always used to convey an IED payload to the mall and its wait-time at the depot is constant. All vehicle sizes deliver to the mall in the benign scenario and their wait-times at the depot are proportional to

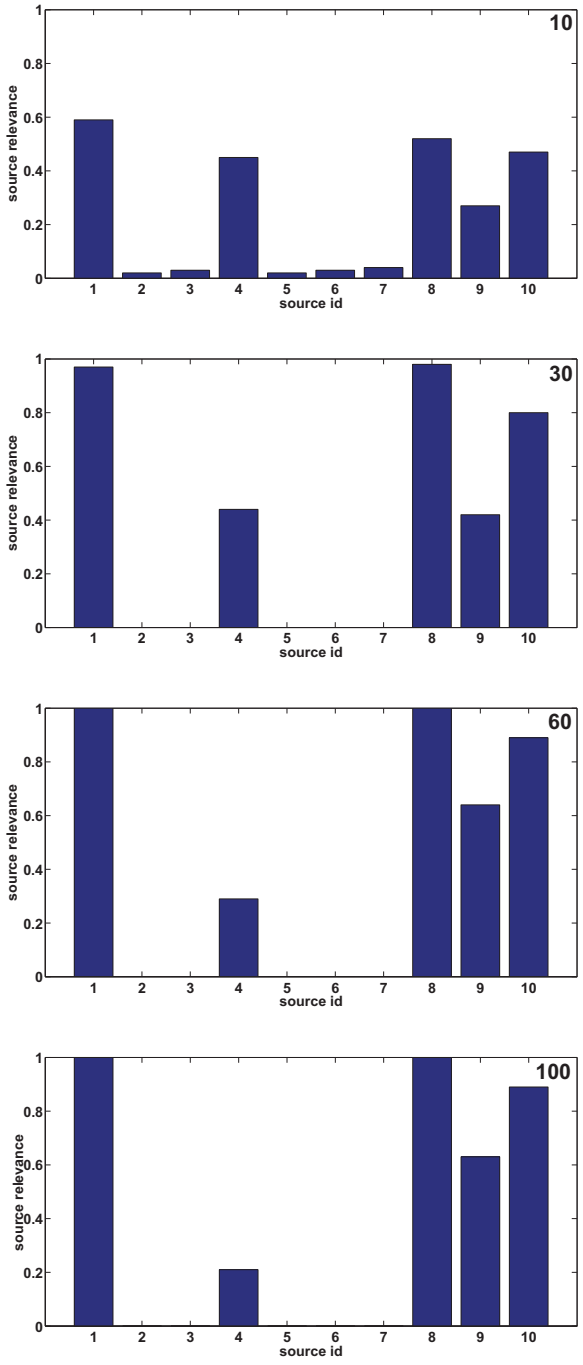

Figure 3. Source relevance probabilities inferred by our framework for different numbers of training inputs (indicated in the top right corner of each figure). The (unknown) true relevant sources in the scenarios that were simulated are 1, 8, 9 & 10.

their size. The observables were generated with uncertainty to blur these subtle distinctions between the classes even further. It would be very difficult for an analyst to determine the relevant sources of data by visual inspection alone.

The relevance probability determined by our framework for each source is displayed in Figure 3 for different numbers of labelled input scenarios. This figure shows the proportion of training instances where each source was identified as relevant.

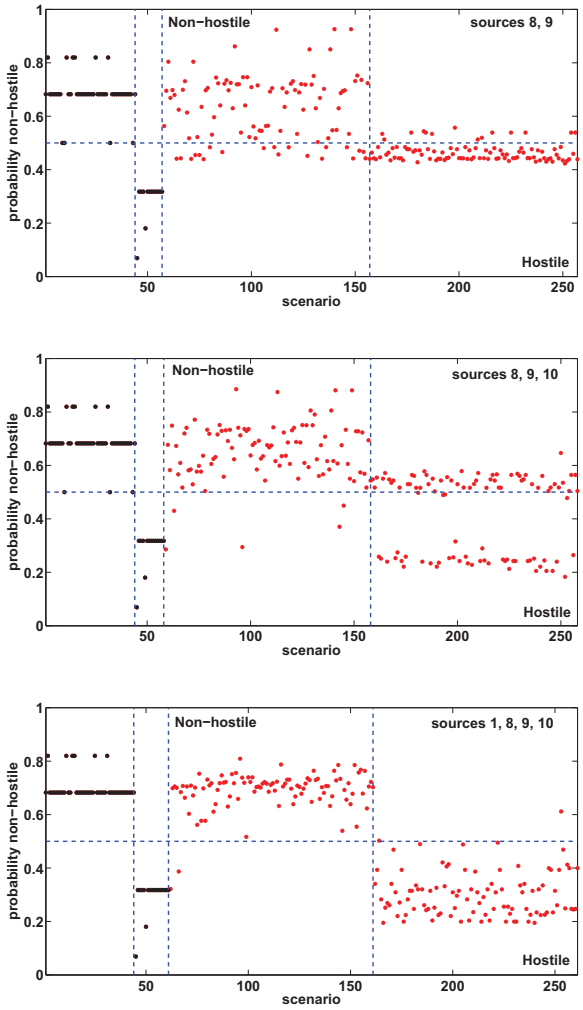

Figure 4. Improved classification performance as a result of fusing more relevant sources

Each training instance uses a randomly selected subset of all possible scenarios. The figure shows that for small training sets (numbering about 30 or less) relevance can vary significantly. However, as the number of training examples increases the relevance probabilities should converge on 0 or 1. The true relevant sources in this example are 1, 8, 9 & 10. The most relevant sources (1 & 8) have indeed converged and the relevance of the other sources is trending in the right direction. Clearly, this performance depends on the amount of overlap in the input scenarios but is a promising indication that relevance detection is effective under challenging (albeit simulated) input conditions.

The next experiments analyse the classification performance for our heterogeneous data fusion framework. The aim is to show how its performance depends on (a) the number and relevancy of information sources used for classification, and (b) the number of scenario datasets used to train the classifier.

The first experiment demonstrates how performance is improved as more relevant sources are fused into the classifier. For this experiment 60 scenarios with a hostile:non-hostile

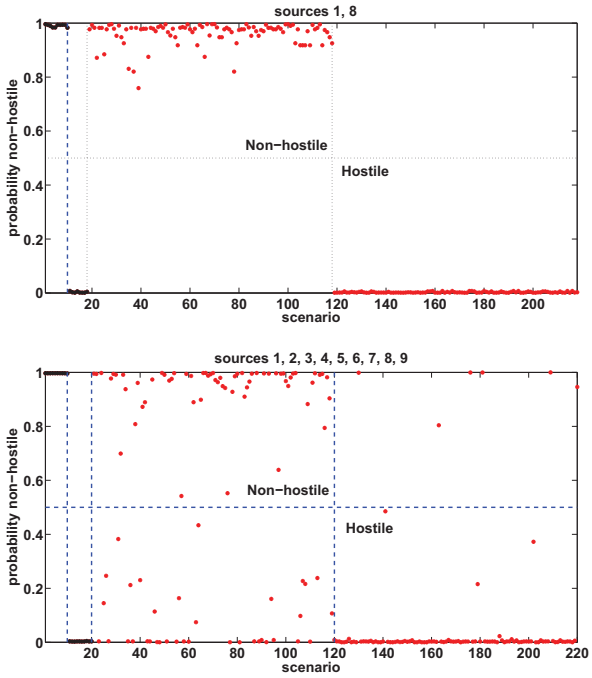

Figure 5. Improved classification performance as a result of excluding irrelevant sources.

ratio of 1:3 were used to train the classifier and 200 scenarios with a true hostile:non-hostile ratio of 1:1 were used to test the classifier.

Fig. 4 displays representative results for the classification of the training and test scenarios when two, three, and four (all) of the relevant vectors are fused in the classifier. These figures plot, for each scenario numbered on the horizontal axis, the probability of being classified non-hostile on the vertical axis. Ignoring the classification of the training scenarios (left of scenario 60), which is nevertheless a useful indicator of the completeness of the training set, the classification of the test scenarios (right of scenario 60) should fall into two blocks on the leading diagonal. The proportion of off-diagonal (misclassified) scenarios is notably reduced as more relevant sources are fused by the classifier.

The next experiment investigates the importance of excluding irrelevant sources from the classifier. The setup is similar to the previous experiment but with slightly less noise on the sources to emphasise performance differences. Figure 5 displays the classification results in the case when two relevant features are fused and also when another relevant feature and five irrelevant features are combined. For this simulated test dataset there are no misclassifications when only the two relevant features are combined. However, inclusion of the irrelevant features (despite the inclusion of an extra relevant feature) introduces numerous misclassifications. This highlights the degradation in classifier performance that is avoided by the ability of our framework to automatically detect the relevant sources and use only them as the basis for generating classifications.

The final experiment analyses the classfier performance for

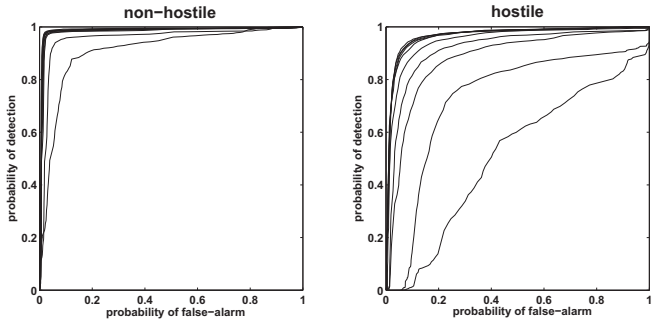

Figure 6. ROC performance results for classifying the non-hostile and hostile scenarios with increasing number of training examples. The lowest ROC curves are for 10 training examples and the highest are for 100 training examples.

different training dataset sizes. Specifically, Receiver Operator Characteristic (ROC) curves are generated by training the classifier against $10, 20, \ldots, 100$ training scenario datasets, with a hostile:non-hostile ratio of 1:4, and testing on 200 scenarios (1:1 ratio). The ROC results for classifying the hostile and non-hostile scenarios in the test dataset are displayed in Fig. 6. These results highlight the expected trend toward lower classification error as the number of training examples is increased. Further work is planned to explore the sensitivity of the results to the underlying scenarios.

7. Conclusions

A novel kernel-based approach for heterogeneous (hard and soft) data fusion has been developed and applied to a synthetic COIN problem. Kernels allow disparate information types to be represented in a common covariance space. The Gaussian Process kernel method provides a rigorous probabilistic basis on which to manage uncertainty and as a side-effect allows relevant sources to be automatically detected.

Important areas for further research would include the following:

- Connect the kernels more directly to features, such as 'bag of words' for documents, and consider distance metrics on those feature spaces
- Consider how to scale-up the computation to handle large (thousands) of training samples
- Consider how to relax the tacit assumption made here that the relevance of sources is stationary throughout a period of surveillance
- Consider how to visualise the outputs of the fusion processes to help intelligence analysts interpret the results and improve decision-making

References

[1] D. L. Hall and J. J. Jordan, *Human-Centered Information Fusion*. Artech House, 2010.

[2] D. L. Hall, R. Nagi, J. Llinas, J. Lavery, and A. Shirkhodaie, "Multidisciplinary research in hard and soft information fusion."

[3] M. A. Pravia, R. Prasanth, P. Arambel, C. Sidner, and C. Y. Chong, "Generation of a fundamental data set for hard/soft information fusion," in *Proceedings of the 11th International Conference on Information Fusion (Fusion 2008), Cologne*, 2008.

[4] S. Das, *High-Level Data Fusion*. Artech House, 2008.

[5] B. Khaleghi, A. Khamis, and F. Karray, "Random finite set theoretic based soft/hard data fusion with application for target tracking," in *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2010 IEEE Conference on*, sept 2010, pp. 50–55.

[6] S. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," *International Journal of Mathematical Models and Methods in Applied Sciences*, vol. 1, no. 4, pp. 300–307, 2007.

[7] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[8] D. Mackay, "The evidence framework applied to classification networks," *Neural Computation*, vol. 4, no. 5, pp. 720–736, 1992.

[9] R. M. Neal, *Bayesian Learning for Neural Networks*, ser. Lecture Notes in Statistics 118. New York: Springer, 1996.

[10] D. Grande, G. Levchuk, W. Stacy, and M. Kruger, "Identification of adversarial activities: profiling latent use of facilities from structural data and real-time intelligence," in *Proceedings of the 13th International Command and Control Research and Technology Symposium, Seattle, WA*, 2008.