Robust Sequential Classification of Tracks

Nathan Parrish, Hyrum Anderson, and Maya R. Gupta
Dept. of Electrical Engineering
University of Washington
Seattle, USA
nparrish@u.washington.edu, hyrum@u.washington.edu, gupta@ee.washington.edu

Abstract − *We present a robust probabilistic method to classify targets based on their tracks. As is customary in supervised learning problems, it is assumed that example tracks from various classes are available to train a classifier. We present an optimal but computationally intensive sequential solution, and show that a computationally feasible naive Bayes approximation works better than ignoring sequential information. We show how to take into account the uncertainty of the track, as quantified by the error covariance matrix from a Kalman tracker, using the recently proposed expected maximum likelihood rule coupled with a robust local Bayesian discriminant analysis classifier. In addition, we propose an expected maximum a posterior rule to take test sample uncertainty into account for classifiers that model the posterior, and use it to define a robust kernel classifier. Simulations with a Kalman tracker show significantly improved performance by taking into account the tracked state covariance.*

Keywords: tracking, classification, Bayesian, quadratic discriminant analysis

1 Introduction

Tracking provides a set of sequential estimates of the state of a dynamic object, usually position and velocity. We study the problem of classifying a test track given example tracks from objects of different classes. Classification of dynamic objects arises in a number of applications and with a variety of sensor modalities, such as identifying friends from foes on the battlefield using sonar or radar measurements [1], separating teammates on the sports field with video tracking [2], or distinguishing between humans and dogs using laser scans [3]. For some applications, classification can be performed independently from tracking by using different data such as sonar signal signatures or imagery data. In this paper, we focus on discriminating objects using only the tracked states. The tracking information may be the only data available, or classification estimates from tracking data can be fused with estimates based on complementary information.

We classify a dynamic object (also referred to as *target*) based upon its state estimates and covariances, as produced by a standard Kalman tracker. We consider two approaches: either classifying based only on the current state, or classifying optimally based on the history of states. Unfortunately, optimal sequential classification is computationally impractical, and we propose a naive Bayes approximation that is simple to use.

A key contribution of this paper is showing that we can greatly improve the classifier performance by taking into account the uncertainty in the Kalman estimate. We do so by applying the recently proposed expected maximum likelihood decision rule applied to a generative classifier [4]. In addition, we introduce an expected maximum a posterior decision rule and use it to create a robust kernel smoothing classifier. In this work we assume that the training tracks are legitimate examples of their class, that is, we do not model the training data as noisy. Simulations show that taking into account the uncertainty in the track estimates can significantly improve classification results.

2 Related Work

Given a sequence of training track states for each class, and a sequence of test track states, one can apply any standard classifier to the feature vector formed by the sequential tracks. Alternatively, a hidden Markov model can be used with class-conditional probabilistic transitions to model the state dynamics.

Here, we focus on incorporating the known uncertainty in the test sample given by the tracker's measurement noise covariance. Related work in dealing with measurement uncertainty when classifying tracks is the work of Luber et al. [3]. They focus on classifying tracks of bearing and range measurements from laser scan data, but their approach is fairly general. They convert each test track into a likelihood over a discrete grid, using a Gaussian distribution to represent the sen-

sor noise. Then a variant of k-means is used to cluster the training track likelihood grids to form exemplar likelihood grids for each class. Transition probabilities between the class exemplars are learned to model the dynamics over time. They classify a test track based on its maximum a posterior probability, using a smoothing kernel probability estimator, and taking into account the state dynamics as modeled by the transition probabilities between the exemplars.

Other models with different assumed class-conditional information have been shown to be useful for classifying tracks. For example, Coraluppi and Carthel use the track length to differentiate targets from clutter [5]. Many researchers have worked on classifying tracks assuming one has *a priori* different kinematic models for each class, and then classifying based on the likelihood of the estimated state dynamics under the different class-conditional kinematic models. For example, Leung and Wu model the maximum acceleration as different for passenger planes and fighter jets [6]. A general approach is to model the state transition matrices as class-conditional, such that the state s_k at time k depends on the past state and on the class g through class-conditional state-transition matrices F_g, G_g and possibly class-conditional noise $w_{k,g}$:

$$s_k = F_g s_{k-1} + G_g w_{k,g}. \tag{1}$$

Optimal Bayesian joint tracking and classification for that framework have been worked out by others [7–9]. That work differs from this research in that they assume that distinct class-specific state transition models are given. We take a more empirical approach, in which a generic state transition model is used for every class, and differences between the models are attributed to modeling noise.

3 Background

To classify a target, we adopt the supervised learning paradigm, in which it is assumed that a set $\mathcal{X} = \{(s_i, g_i)\}_{i=1}^{M}$ of labeled feature vectors is available to train a classifier. We take $s_i \in \mathbb{R}^d$ to be a valid state of a target belonging to class $g_i \in \mathcal{G}$. Having trained a classifier from the training data, we may then classify a target based on its true state s.

In this section, we review recent work on robust classification, from which we build classifiers for Kalman tracks. First, in Sec. 3.1, we review the expected ML rule for classifying noisy test samples. Then, in Sec. 3.2, we review the robust Bayesian quadratic discriminant analysis classifier.

3.1 Expected ML rule for Noisy Features

One traditional statistical approach to classify s from $\mathcal{X}$ is the maximum likelihood (ML) classification rule. First, we model s as a realization of random state S with likelihood $p(s|g)$ inferred from the training set $\mathcal{X}$. Then, if given the true state s of a target, the ML rule is to classify the target as the class that solves $\arg\max_g p(s|g)$.

However, it is most often the case that the true state s of the target is not observed directly. Rather, we observe a noisy measurement z of s. Sequential filtering and tracking methods (Kalman filter, particle filter, etc.) estimate the expected state and error covariance of the (unknown) true state of a target using noisy and possibly nonlinear measurements z at discrete time intervals.

Anderson and Gupta recently proposed the expected ML rule to robustly classify noisy observations [4], and in this paper we apply the expected ML rule for classifying target tracks via the Kalman filter. The authors showed in experiments that the expected ML rule can tolerate moderate amounts of measurement noise, and that it far outperforms the naive approach of simply treating the estimate $\hat{s}$ as the true feature vector. Given a noisy measurement z of S, the expected ML rule prescribes classifying z as class

$$\hat{g} = \arg\max_{g \in \mathcal{G}} \quad \mathrm{E}_{S|z}\left[p(S|g)\right] p(g), \tag{2}$$

$$\text{where} \quad \mathrm{E}_{S|z}\left[p(S|g)\right] = \int p(s|z) p(s|g)\, ds, \tag{3}$$

where the prior $p(g)$ accounts for unbalanced classes, also inferred from the data[1].

In general, the integral in (3) must be solved numerically. However, when both $p(s|z)$ and $p(s|g)$ can be represented as Gaussian distributions, then the product of Gaussians rule can be used to derive a closed form solution [4]: let $p(s|z) = \mathcal{N}(s; a, A)$ and $p(s|g) = \mathcal{N}(s; b, B)$; then,

$$\begin{aligned} &\mathcal{N}(s; a, A)\mathcal{N}(s; b, B) \\ &\qquad = \mathcal{N}(a; b, A + B)\mathcal{N}(s; c, C), \end{aligned} \tag{4}$$

where $C = \left(A^{-1} + B^{-1}\right)^{-1}$ and $c = C(A^{-1}a + B^{-1}b)$. Substituting (4) into (2), the term $\mathcal{N}(a; b, A + B)$ on the right hand side of (4) can be moved outside the integral, and the term $\mathcal{N}(s; c, C)$ integrates to unity. This results in a closed form solution for (2):

$$\hat{g} = \arg\max_g \mathcal{N}(a; b, A + B).$$

Modeling the distribution $p(s|z)$ as Gaussian arises naturally from the Kalman filter, where the mean a and covariance A are taken to be the estimate and error covariance matrix. Following the supervised learning paradigm, the class-conditional distribution $p(s|g)$

[1] Anderson and Gupta call their rule "expected MAP rule" due to the inclusion of the prior $p(g)$; however, to avoid confusion, we refer to it as the expected ML rule, since expectation is performed only on the likelihood.

can be modeled as Gaussian with parameters derived from the training data $\mathcal{X}$. In section (3.2), we detail a classification method that models $p(s|g)$ as Gaussian, and describe how to robustly estimate the parameters of the class-conditional distribution.

3.2 BDA and Robust BDA Classifiers

Quadratic discriminant analysis (QDA) models the samples from each class as being drawn independently and identically from a Gaussian distribution [10]. This simple model is usually not flexible enough to capture the true distribution, and a standard practice is to use a Gaussian mixture model (GMM) [10]. For example, GMMs were found to work well for classifying radar tracks as having been generated from planes vs. birds, compared to a support vector machine, neural network, and single-Gaussian-per-class models [11].

However, learning a GMM can be computationally expensive and lead to overfitting, so in this work we use a simpler alternative that was recently proposed called local QDA [4] [12]. Local QDA fits one Gaussian model per class, but only to the k nearest neighbors of a test sample, where k is a parameter that can be chosen or averaged over [12]. Local QDA can also be viewed as a generalization of the local nearest means classifier [13]. To avoid estimation problems when $k < d$, the local Bayesian QDA (local BDA) classifier uses data-dependent Bayesian estimation for each class-conditional Gaussian (see [14] for details on BDA estimation). The resulting local BDA classifier has been shown to produce state-of-the-art classification results [12].

Anderson and Gupta extended the BDA classifier for use with noisy test samples. The robust BDA (R-BDA) classifier approximates the BDA class-conditional distribution $p(s|g)$ as a Gaussian, and employs the expected ML rule in (2) to form a classifier of the form $\hat{g} = \arg\max_g \mathcal{N}(s; \bar{s}_g, \hat{\Sigma}_s)$, where $\bar{s}_g$ is the class-conditional sample mean and $\hat{\Sigma}_s$ is the covariance matrix specified by the BDA class-conditional distribution (see [4] for details).

4 Expected MAP Rule and Robust Kernel Classifier

Some classifiers model the posterior distribution $p(g|s)$ directly, rather than modeling $p(s|g)$. For example, kernel smoothers model the posterior as [10]:

$$p(g|s) \propto \sum_{i=1}^{M} I_{g_i=g} K(s, s_i),$$

where the kernel function $K(s, s_i)$ is often chosen to be a radial function such as the Gaussian radial basis function $K(s, s_i) = \mathcal{N}\left(s; s_i, \gamma^{-1} I\right) \propto \exp\left(-\frac{\gamma}{2}\|s - s_i\|^2\right)$. The kernel rule is used by maximizing the posterior $p(g|s)$ directly given a test sample s.

For such classifiers, analogous to the expected ML rule in (2), we introduce the expected MAP rule for noisy data, which solves

$$\hat{g} = \arg\max_g \quad E_{S|z}[p(g|S)]. \tag{5}$$

For a kernel smoother, this becomes

$$\hat{g} = \arg\max_g \sum_{i=1}^{M} I_{g_i=g} \int p(s|z) K(s, s_i)\, ds. \tag{6}$$

When $K(s, s_i)$ is chosen to be the radial basis function with bandwidth parameter γ, and the distribution $p(s|z) = \mathcal{N}(s; a, A)$, the noise-robust kernel (R-kernel) becomes

$$\mathrm{E}_{S|z}\left[K(S, s_i)\right] = \mathcal{N}\left(s_i; a, A + \gamma^{-1} I\right).$$

We note that an earlier kernel classifier for an additive noise model $\hat{s} = s + \epsilon$ was proposed in [15] that weights training points by the noise distribution $p(\hat{s}|s_i)$,

$$\hat{g} = \arg\max_g \sum_i I_{g_i=g}\, p(\hat{s}|s_i) K(\hat{s}, s_i).$$

We use Pawlak's smoothing kernel to account for arbitrary uncertainty given by the distribution $p(\hat{s}|s_i)$. We note that if there is no uncertainty in the test point, then the covariance matrix $A = 0$; therefore, the Gaussian probability evaluates to zero at all points other than the training point s_i. In practice, this can pose computational problems for even small covariance.

5 Kalman Filtering Single and Sequential Approaches

In this section, we briefly describe the operation of the Kalman filter as it relates to tracking dynamic targets. We then describe how the robust classification methods from the previous sections can be used to classify the noisy state estimate at the output of the Kalman filter.

The linear Kalman filter provides a minimum mean square error estimate of a target state vector when both the dynamic and measurement models are linear with zero mean independent Gaussian noise. In the discrete time filter, the target state vector at time index k is given as s_k, and consists of the target kinematic information. The state dynamics and measurement models are given as:

$$s_k = F s_{k-1} + D u_{k-1} + G w_{k-1} \tag{7}$$

$$z_k = H s_k + v_k \tag{8}$$

where z_k and u_k are the measurement and (known) control input at step k. The matrices F and G are used to model the dynamics of the target, and the observation

matrix H encapsulates the measurement process. Uncertainty is represented by w_k and v_k, which are zero mean Gaussian random vectors with covariance matrices Q and R, respectively.

Suppose that at step $k-1$ we have estimated the mean $\hat{s}_{k-1}$ and error covariance P_{k-1} of s_{k-1}. The Kalman filter performs two steps to form the estimate $\hat{s}_k$ from the previous state estimate: the prediction step and the update step. The prediction step predicts the state vector at time k using the dynamic model and the estimate at time $k-1$:

$$\begin{aligned}\hat{s}_k^- &= F\hat{s}_{k-1} + Du_{k-1}\\ P_k^- &= FP_{k-1}F^T + GQG^T.\end{aligned}$$

The update step utilizes the measurement z_k to update the predicted state estimate:

$$\hat{s}_k = \hat{s}_k^- + K_k[z_k - H\hat{s}_k^-] \tag{9}$$

$$P_k = [I - K_kH]P_k^-, \tag{10}$$

where K_k is the Kalman gain matrix [16].

Equations (9) and (10) allow us to model $p(s_k|z^k)$ (where the superscript denotes all measurements up to time k, that is $z^k = [z_1^T \ldots z_k^T]^T$) as conditionally Gaussian with mean $\mathrm{E}[s_k|z^k] = \hat{s}_k$ and $\mathrm{cov}[s_k|z^k] = P_k$. We can, therefore, classify the target at state k using the expected ML rule given in (2), the R-kernel rule (6), or the Pawlak smoothing kernel (4).

The above methodology allows us to classify a single state vector; however, it would be useful to form a sequential classification rule that can update the class label as new measurements from the target arrive. Let s^k denote a vector containing the target state information at steps 1 through k. The optimal MAP classification rule is then:

$$\arg\max_{g\in\mathcal{G}} \quad p(s^k|g)p(g). \tag{11}$$

As per (7), the state dynamics are a Markov process such that $p(s^k|g) = p(s_1|g)\prod_{i=2}^k p(s_i|g, s_{i-1})$. Therefore, at step k (11) becomes:

$$\begin{aligned}\hat{g}_k &= \arg\max_{g\in\mathcal{G}} p(g)p(s_1|g)\prod_{i=2}^k p(s_i|g, s_{i-1}) \qquad (12)\\ &= \arg\max_{g\in\mathcal{G}} p(g)p(s_k|g, s_{k-1})p(s^{k-1}|g),\end{aligned}$$

which can be computed recursively.

Converting (12) into the corresponding expected ML rule produces:

$$\hat{g}_k = \arg\max_{g\in\mathcal{G}} \mathrm{E}_{S^k|z^k}\left[p(g)p(s_1|g)\prod_{i=2}^k p(s_i|g, s_{i-1})\right] \tag{13}$$

$$\equiv \arg\max_{g\in\mathcal{G}} p(g)\int p(s^k|z^k)p(s_1|g)\prod_{i=2}^k p(s_i|g, s_{i-1})\,ds^k$$

We can again take advantage of the Markov property in the dynamic model to rewrite $p(s^k|z^k) = p(s_1|z^k)\prod_{j=2}^k p(s_j|z^k, s_{j-1})$ which we approximate as $p(s_1|z^1)\prod_{j=2}^k p(s_j|z^j, s_{j-1})$ in order to arrive at a sequential solution. Substituting this form of the conditional probability into (13) gives:

$$\hat{g}_k = \arg\max_{g\in\mathcal{G}}\left(p(g)\int p(s_1|z^1)\prod_{j=2}^k p(s_j|z^j, s_{j-1})p(s_1|g) \cdot \prod_{i=2}^k p(s_i|g, s_{i-1})\,ds^k\right). \tag{14}$$

To simplify (14) to a form that can be updated sequentially, we apply the so-called Naive Bayes rule [10] and approximate the state s_k as being independent of any other state when conditioned on g or z^k. Then, we can rewrite (14) as:

$$\begin{aligned}\hat{g}_k &= \arg\max_{g\in\mathcal{G}}\left(p(g)\int\prod_{j=1}^k p(s_j|z^j)\prod_{i=1}^k p(s_i|g)ds^k\right)\\ &= \arg\max_{g\in\mathcal{G}}\left(p(g)\int p(s_k|z^k)p(s_k|g)\,ds_k \cdot \int p(s^{k-1}|z^{k-1})p(s^{k-1}|g)\,ds^{k-1}\right). \qquad (15)\end{aligned}$$

Equation (15) allows an update of the class label estimate when each new measurement arrives. The update consists of solving the integral $\int p(s_k|z^k)p(s_k|g)\,ds_k$ at the kth observation using the product of Gaussians rule in (4), then multiplying the result with the runningtally integral $\int p(s^{k-1}|z^{k-1})p(s^{k-1}|g)\,ds^{k-1}$ that was updated at step $k-1$.

6 Simulation Setup

We evaluate the performance of the proposed classification methods by simulating a tracking system that tracks targets from two possible classes. Targets from both class 1 and class 2 are generated using the dynamics in equation (7). We simulate a target moving in a 2-dimensional plane, therefore, the state vector $s_k = [x \quad \dot{x} \quad y \quad \dot{y}]^T$. The F, D, G matrices are identical for each class with:

$$F = \begin{bmatrix}1 & T & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & T\\ 0 & 0 & 0 & 1\end{bmatrix} \quad D = G = \begin{bmatrix}T^2/2 & 0\\ T & 0\\ 0 & T^2/2\\ 0 & T\end{bmatrix} \tag{16}$$

where $T = 0.25$ is the sampling interval.

The two classes are distinguished by the input vector u_k, the measurement noise covariance Q and the initial state s_0. The values of these parameters for each class are given in table 1. Note that the initial x position and velocity are the same for each class, whereas the initial y position and velocity are distributed as Gaussian

	Class 1	Class 2
u_k	$[0 \quad 0.1]^T$	$[0 \quad -0.1]^T$
Q	$\begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$	$\begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$
x_0	-50	-50
$\dot{x}_0$	10	10
y_0	$\mathcal{N}(52, 3)$	$\mathcal{N}(48, 3)$
$\dot{y}_0$	$\mathcal{N}(-0.4, 0.3)$	$\mathcal{N}(0.4, 0.3)$

Table 1: Parameters for Class 1 and 2

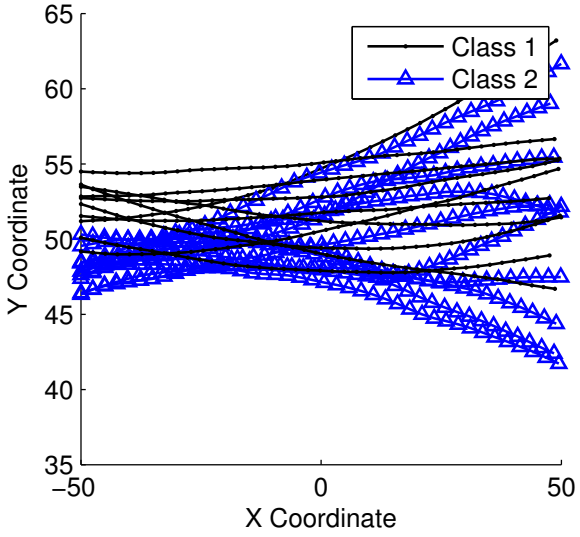


Figure 1: Sample paths for class 1 and 2 tracks. The targets move from the left side of the region to the right side.

random variables that differ between classes by their means. Figure 1 plots ten sample paths for each of the classes. Note that, although the underlying dynamics of each target is Gaussian, $p(s_k|z^k)$ is not Gaussian, but is rather a Gaussian mixture. The Kalman tracker makes a Gaussian assumption for this probability which we use for classification.

We track the classes using a Kalman filter with the F and G matrices given in (16). However, our dynamic model for filtering does not include any control input. The filter is initialized with $\hat{s}_0 = [-50 \quad 10 \quad 50 \quad 0]^T$ and $P_0 = \text{diag}[2 \quad 2 \quad 4 \quad 1.5]$. Additionally, the measurement model and measurement noise covariance matrix are given as:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} \sigma_m^2 & 0 \\ 0 & \sigma_m^2 \end{bmatrix}. \qquad (17)$$

Figure 2 shows the error signal between the true and estimated state, as well as the three sigma bounds of the error variance for each signal. We can see that the measurement error variance decreases rapidly over the first few iterations, and then converges to a stable value.

We compare five different classifiers: BDA, R-BDA, a standard Gaussian kernel classifier, the R-kernel, and the Pawlak smoothing kernel. BDA and the standard kernel are 'non-robust,' in that they consider the estimated state $\hat{s}_k$ as the true state and ignore the error covariance.

We generate 100 clean tracks for training. The tracks are generated such the classes are equally likely: $p(g_1) = p(g_2) = 0.5$. The feature vectors in the training data are the clean states segregated by the state index k. In other words, $p(s_k|g)$ is trained only using the labeled training set of 100 k index states, or in the case of the kernel classifier, the test state k is only compared to training states also with index k.

The parameters for the classifiers are learned by 5-fold cross validation. In each cross validation run, the cross validation test set is made to look like the actual test set by measuring the cross validation test states according to the measurement equation with appropriate H and R matrices. These noisy measurements are propagated through the same Kalman filter that is used to generate the real test data so that each test set used in cross validation has the same statistics as the actual test data.

We perform simulations with five different values of measurement noise covariance σ_m^2 using the same test and training sets. Note that in the case of $\sigma_m^2 = 0$, both the test and training sets have no uncertainty. The Monte Carlo simulation is performed twice, with new test and training sets generated on each run.

7 Results and Discussion

Figure 3 shows the error rate versus the measurement noise variance for both the single state classification method and the sequential method using the naive Bayes assumption. The results at each value of measurement noise variance are averaged over all sampling indices. We can see that the noise robust methods, R-BDA and R-kernel, significantly outperform their non-robust counterparts in both sets of simulations. By comparing figures 3 (a) and (b), we can see that sequential R-BDA and R-Kernel outperform their non-sequential counterparts at all noise levels. On the other hand, the non-robust methods perform worse in the sequential classifiers than in the non-sequential at all measurement noise levels other than $\sigma_m^2 = 0$ and $\sigma_m^2 = 0.1$.

Figure 4 plots the error rate vs. the sampling index for a fixed measurement noise covariance of 1 where (a) and (b) show the single state method and sequential method, respectively. This figure gives a clear interpretation of why the robust classifiers perform better in sequential classification than single state at all levels of noise covariance; whereas the non-robust classifiers perform worse with sequential classification as the noise covariance increases. We can see in figure 4 (a) that for

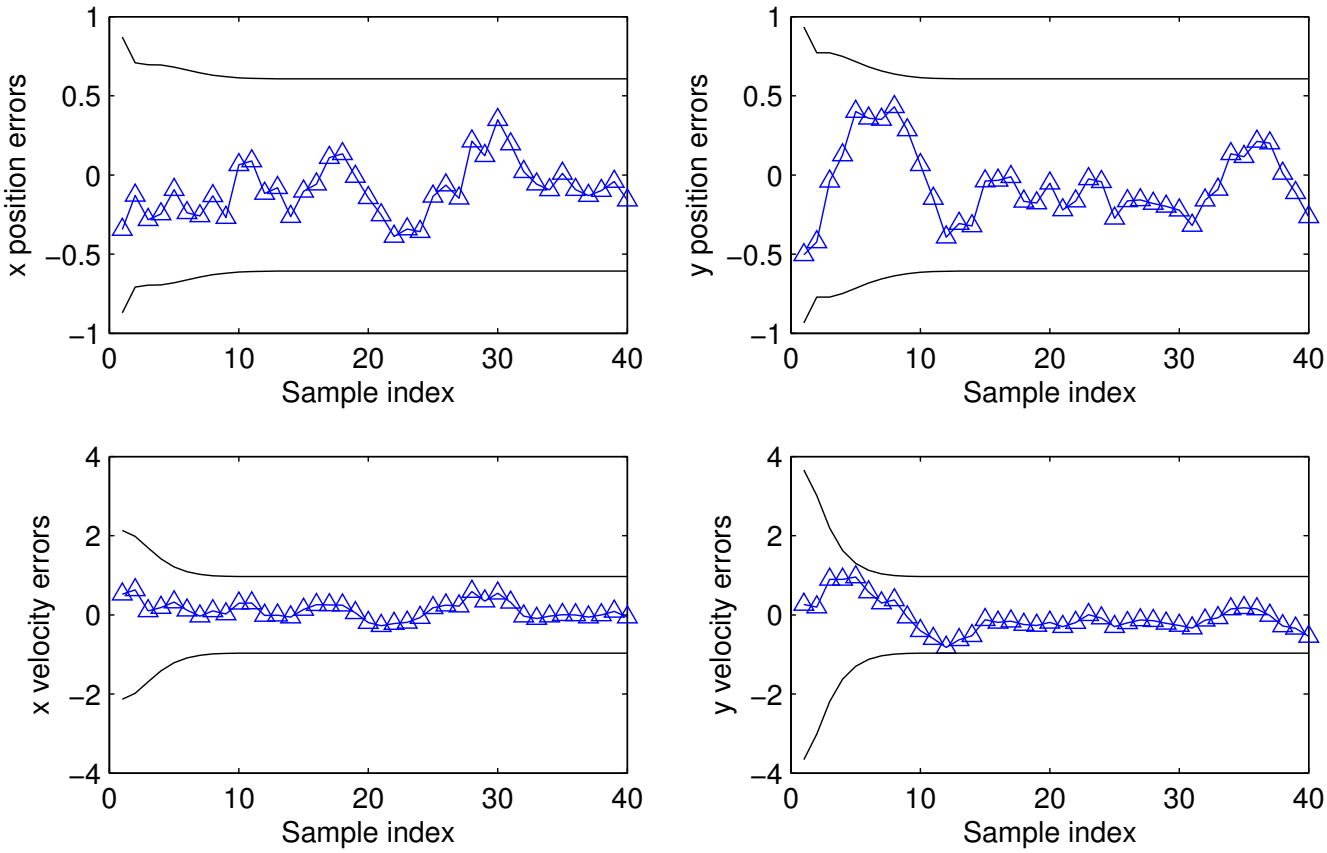


Figure 2: Error signal and 3 sigma bounds on the error variance for the four components of the state vector as a function of sampling index k. The measurement noise variance is $\sigma_m = 0.1$.

single state estimation the non-robust classifiers perform poorly when the state indices are small ($k < 10$). Referring to figure (2) shows that this is when the uncertainty in the true tracks is greatest. However, the robust classifiers are able to make much better single state estimates even under this uncertainty, and thus have the worst performance when $20 < k < 30$. Referring to 1), this is when the tracks exhibit the most mixing. The effect of these observations on the sequential classification error rate in figure (4) (b) is that the robust estimation methods are able to leverage the good information that they learn at small k to aid in classification at larger k. However, the non-robust methods are hampered by poor decisions early that carry over into larger k.

We also note that in figure 3, the Pawlak smoothing kernel performs worse than the non-robust kernel method at measurement noise variance $\sigma_m^2 = 0.1$. This is due to the computational difficulties that we addressed in section 4. In the case of no test point uncertainty, we edited the Pawlak smoothing kernel to be equivalent to the standard Gaussian kernel.

8 Conclusions

We have presented a sequential method for classifying tracks under uncertainty. We have shown that the noise robust methods significantly outperform the non-robust methods, particularly in the case of the naive Bayes sequential classifier.

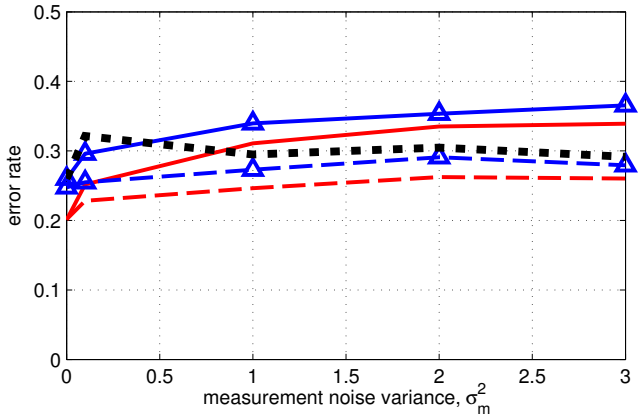
References

[1] L. D. Stone, T. L. Corwin, and C. A. Barlow, *Bayesian Multiple Target Tracking*. Artech House Publishers, 1999.

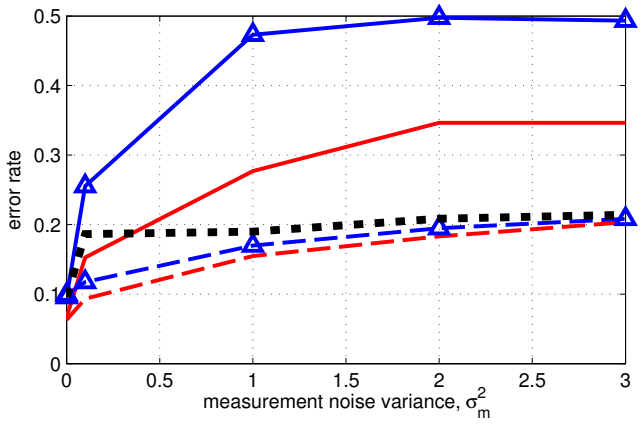
[2] D. Thirde, M. Xu, and J. Orwell, *Intelligent distributed video surveillance systems*. IEE Professional Applications of Computing Series, 2006, vol. 5, ch. Tracking football players with multiple cameras, pp. 225–252.

[3] M. Luber, K. O. Arras, C. Plagemann, and W. Burgard, "Classifying dynamic objects: An unsupervised approach," *Autonomous Robotics*, vol. 26, pp. 141–151, 2009.

[4] H. S. Anderson and M. R. Gupta, "Classifying linear system outputs by robust local bayesian



(a) Single State



(b) Naive Bayes Sequential

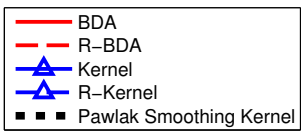
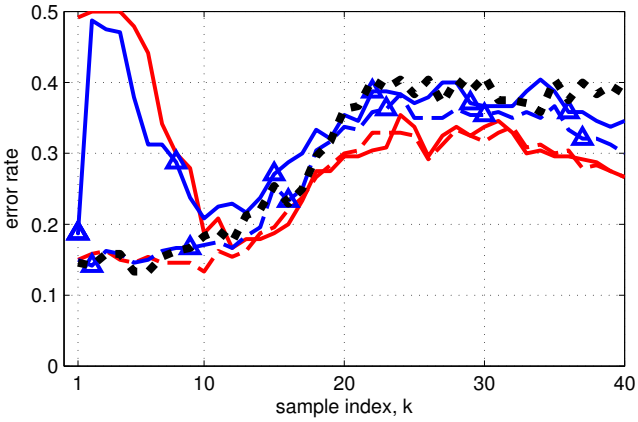
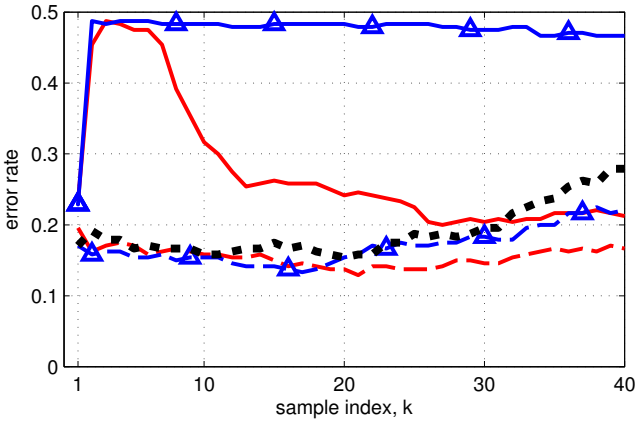


Figure 3: Error rate vs. measurement noise covariance σ_m^2. The results show the error averaged over all state indices.



(a) Single State



(b) Naive Bayes Sequential

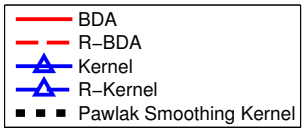


Figure 4: Error rate vs. state index k for fixed measurement noise covariance $\sigma_m^2 = 1$.

quadratic discriminant analysis on linear estimators," *Proc. IEEE Workshop on Statistical Signal Processing*, 2009.

[5] S. Coraluppi and C. Carthel, "Multi-hypothesis sonar tracking," *Proc. Intl. Conf. Information Fusion*, 2004.

[6] H. Leung and J. Wu, "Bayesian and Dempster-Shafer target identification for radar surveillance," *IEEE Trans. Aerospace and Electronic Systems*, vol. 36, no. 2, pp. 432–447, 2000.

[7] B. Ristic, N. Gordon, and A. Bessell, "On target classification using kinematic data," *Information Fusion*, vol. 5, pp. 15–21, 2004.

[8] N. J. Cutaia and J. A. O'Sullivan, "Automatic target recognition using kinematic priors," *Proc. Conf. Decision Control*, pp. 3303–3307, 1994.

[9] S. Challa and G. W. Pulford, "Joint target tracking and classification using radar and ESM sensors," *IEEE Trans. Aerospace and Electronic Systems*, vol. 37, no. 3, pp. 1039–1055, 2001.

[10] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York: Springer-Verlag, 2001.

[11] L. P. Espindle and M. J. Kochenderfer, "Classification of primary radar tracks using gaussian mixture models," *IET Radar, Sonar, Navigation*, vol. 3, no. 6, pp. 559–568, 2009.

[12] E. K. Garcia, S. Feldman, M. R. Gupta, and S. Srivastava, "Completely lazy learning," *IEEE Trans. Knowledge and Data Engineering*, 2009, published Early Access, Print Version To Appear.

[13] Y. Mitani and Y. Hamamoto, "A local mean-based nonparametric classifier," *Pattern Recognition Letters*, vol. 27, pp. 1151–1159, 2006.

[14] S. Srivastava, M. R. Gupta, and B. A. Frigyik, "Bayesian quadratic discriminant analysis," *J. Mach. Learn. Res.*, vol. 8, pp. 1287–1314, 2007.

[15] M. Pawlak and D. Siu, "Pattern classification with noisy features," *Lec. Notes in Comp. Sci.*, vol. 1451, pp. 845–852, 1998.

[16] J. L. Crassidis and J. L. Junkins, *Optimal Estimation of Dynamic Systems*. CRC Press, 2004.