SCALABLE ROBUST HYPOTHESIS TESTS USING GRAPHICAL MODELS

Divyanshu Vats†, Vishal Monga‡, Umamahesh Srinivas‡, José M. F. Moura†

†Dept. of Elec.and Comp. Eng., Carnegie Mellon University, Pittsburgh, PA, 15213
‡Department of Electrical Engineering, Pennsylvania State University, University Park, PA 16802

ABSTRACT

Traditional binary hypothesis testing relies on the precise knowledge of the probability density of an observed random vector conditioned on each hypothesis. However, for many applications, these densities can only be approximated due to limited training data or dynamic changes affecting the observed signal. A classical approach to handle such scenarios of imprecise knowledge is via minimax robust hypothesis testing (RHT), where a test is designed to minimize the worst case performance for all models in the vicinity of the approximated imprecise density. Despite the promise of RHT for robust classification problems, its applications have remained rather limited because RHT in its native form does not scale gracefully with the dimension of the observed random vector. In this paper, we use approximations via probabilistic graphical models, in particular block-tree graphs, to enable computationally tractable algorithms for realizing RHT on high-dimensional data. We quantify the reductions in computational complexity. Experimental results on simulated data and a target recognition problem show minimal loss over a true RHT.

1. INTRODUCTION

Recognizing the sensitivity of classical Neyman-Pearson and Bayesian hypothesis tests, Huber [1] proposed a robust hypothesis test (RHT) that is insensitive to errors in modeling densities of each hypothesis. The motivation for using such tests is clear from practical situations in which the true distribution of each hypothesis can be only approximated due to limited/noisy training samples, and/or physical effects on the observed signal that go unmodeled. To account for the uncertainty in the distribution of each hypothesis, Huber designed a minimax RHT that minimized the worst case performance of observing a single observation by assuming that the uncertainty in the distribution of each hypothesis followed a contamination model. Extensions of the RHT framework to other measures of model proximity have been proposed in [2–4]. For a survey on RHT and other robust methods in signal processing, see [5].

In this paper, we consider a problem of high-dimensional RHT, where the observed signal is high-dimensional with imprecise probability densities under each hypothesis and the goal is to design the minimax RHT under the contamination model. Although the solution due to Huber can be directly applied, we show that computing this robust test becomes computationally intractable due to the high-dimensionality of the problem. The main computationally intensive step in computing the RHT involves solving two highly nonlinear equations (9) and (10) using Monte Carlo based methods, which requires random sampling to empirically estimate a probability measure. To counter the computational burden of computing RHT, we approximate the distribution under each hypothesis using a graphical model [6], specifically a block-tree graph [7]. This has two advantages, (i) random sampling on block-tree graphs is computationally efficient, and (ii) the number of samples needed from the approximated distribution for Monte Carlo methods is much less than the number of random samples needed from the original distribution. We quantify the reductions in complexity and show via experimental results on simulated and real (target classification) data that our proposed algorithm only incurs a slight loss in performance over a true RHT (which is intractable).

Although computing a high-dimensional minimax RHT has not been studied before, computing robust tests assuming a certain form for the classifier has been studied in the literature [8, 9]. The RHT problem is more general since the form of the classifier depends on the distribution of each hypothesis. References [10–12] study the use of graphical models in hypothesis testing, however, these works do not design robust tests.

2. ROBUST HYPOTHESIS TESTING (RHT)

Section 2.1 reviews the minimax robust hypothesis tests (RHT). Section 2.2 highlights the computational challenges in computing the minimax RHT for high-dimensional signals.

2.1. Mimimax RHT

Let x = (x1, . . . , xn) be a random vector generated from one of two hypotheses, H0 or H1, where the distributions g(x|H0) and g(x|H1) are unknown. Instead, for each hypothesis, a nominal density fk(x) is known, and the true density is assumed to be in some neighborhood of the nominal density. These neighborhoods can be specified using various metrics for model proximity, see [3] for a comprehensive review. In this paper, we use the contamination model to construct neighborhoods around the nominal density as

\[ \mathcal{F}_k = \{ f(x) : f(x) = (1 - \epsilon_k) f_k(x) + \epsilon_k h(x) \}, \quad k = 0, 1, \]

where 0 ≤ \epsilon_0, \epsilon_1 ≤ 1, and h(x), the contamination, is an unknown probability density function. Let \( \mathcal{F} = \mathcal{F}_0 \times \mathcal{F}_1 \). The contamination model can be used in underwater acoustics to model the effect of unexpected noise sources [3]. We now formulate the minimax RHT problem of Huber [1] using the notations in [3].

Let \( \mathcal{D} \) be a convex set of pointwise randomized decision functions denoted by \( \delta(\cdot) \) such that, for observation y, we select H1 with probability \( \delta(y) \) and H0 with probability \( 1 - \delta(y) \). The probability of false alarm and probability of miss is given as

\[ P_F(\delta, f_0) = \int \delta(y) f_0(y) dy \]

\[ P_M(\delta, f_1) = \int (1 - \delta(y)) f_1(y) dy . \]

For equally likely hypotheses, the probability of error is given by

\[ P_E(\delta, f_0, f_1) = \frac{1}{2} [ P_F(\delta, f_0) + P_M(\delta, f_1) ] . \]
For the RHT, we want to solve the following minimax problem
\[(\delta_R, f_0^L(x), f_1^L(x)) = \arg\min_{\delta \in D} \max_{f_0, f_1 \in F_c} P_E(\delta, f_0, f_1), \]
(5)
where \(\delta_R\) is the robust test and \((f_0^L, f_1^L)\) are least favorable densities in \(F_c\). The robust test \(\delta_R\) performs appropriately not only for a given model, but for an entire class of models in the neighborhood of the nominal density. As shown by Huber [1], the solution to (5) is
\[f_0^L(x) = \begin{cases} 
1 - \epsilon_0 & f_0(x) \geq c'' \nonumber \\
\frac{1}{\epsilon_0} (1 - \epsilon_0) f_1(x) & f_0(x) < c'' 
\end{cases} \]
(6)
\[f_1^L(x) = \begin{cases} 
1 - \epsilon_1 & f_1(x) > c' \\
(1 - \epsilon_1) f_0(x) & f_1(x) \leq c' 
\end{cases} \]
(7)
\[\delta_R(x) = \begin{cases} 
1 & f_0^L(x) > f_1^L(x) \geq 1 \\
0 & f_0^L(x) \leq f_1^L(x) < 1 
\end{cases} \]
(8)
where \(c'\) and \(c''\) are defined such that \(f_0^L\) and \(f_1^L\) are valid probability distributions giving us the equations
\[P_0 \left( \frac{f_1(x)}{f_0(x)} < c'' \right) + \frac{1}{\epsilon_0} P_1 \left( \frac{f_1(x)}{f_0(x)} \geq c'' \right) = \frac{1}{1 - \epsilon_0} \]
(9)
\[P_1 \left( \frac{f_1(x)}{f_0(x)} > c' \right) + c' P_0 \left( \frac{f_1(x)}{f_0(x)} \leq c' \right) = \frac{1}{1 - \epsilon_1} \]
(10)
P_0 is the probability measure w.r.t the nominal density \(f_k(x)\).

2.2. Scalability Challenges
From (9)-(10), it is clear that instantiating RHT is tantamount to determining the parameters \(c'\) and \(c''\). A solution to (9) and (10) exists such that \(c' < c''\) as long as \(\epsilon_0\) and \(\epsilon_1\) are small [1]. However, computing \(c'\) and \(c''\) using (9) and (10) requires the use of Monte Carlo methods, where we generate random samples of \(x\) under both \(f_1(x)\) and \(f_0(x)\) to compute the probability measures in (9) and (10). As the dimension of \(x\) increases, the number of random samples required increases exponentially. For example, if each \(x_k\) is binary, \(x\) can take one of \(2^n\) possible values. This means that at least \(2^n\) number of samples need to be generated for reasonable approximations. Further, generating samples from high-dimensional data is often hard. Recognizing the computational intractability in finding \(c'\) and \(c''\) exactly, we instead find approximations. In particular, we use graphical models to compute these approximations.

3. SCALABLE RHT USING GRAPHICAL MODELS
Graphical models have found many applications such as in sensor networks, image processing, and computer vision owing to their ability to effectively capture data dependencies while simultaneously providing complexity benefits in inference tasks [13]. Section 3.1 outlines our approach to using block-tree graphs, introduced in [7], for approximating distributions by graphical models. Section 3.2 discusses how we use the block-tree approximations in solving for \(c'\) and \(c''\) to compute the RHT. Section 3.3 discusses the computational benefits of using graphical models for computing \(c'\) and \(c''\).

3.1. Block-Tree Graphical Approximations
Given a distribution \(f(x)\), this section outlines an algorithm to find approximate recursive structures amongst random variables in \(x\) that enables efficient random sampling, which is needed to compute \(c''\) and \(c'\) in (9) and (9). We use the framework of block-tree graphs [7], which is defined by a tuple \(G = (C, E)\), where \(C = \{C_1, \ldots, C_r\}\)

is a set of disjoint clusters of nodes and \(E\) is a set of edges between clusters having a tree-structure. An example of a block-tree graph is shown in Fig. 1. A random vector \(x\) is defined on a block-tree graph in such a way that each cluster \(C_i\) corresponds to a random vector \(x_{C_i}\) \((i = 1, \ldots, r)\) and the arrangement of edges between clusters reveals conditional (in)dependencies between the set of random vectors \(\{x_{C_1}, \ldots, x_{C_r}\}\).

The advantage of using block-tree graphs for graphical approximations, as opposed to more general graphs, is that (i) greedy algorithms for computing block-trees are computationally efficient, and (ii) given a block-tree graph, we can derive recursive algorithms for generating samples. To estimate the block-tree graph structure, we use the following two step procedure:

**Algorithm 1:** Estimating Block-Tree Structure

1. **Choose clusters:** Assume \(|C_j| = m\) for all \(1 \leq j \leq r\) and \(|C_j| = n - m(r - 1)\), where \(r = m\). Starting with \(x_1\), find \(m - 1\) nodes that are maximally correlated with \(x_1\). Assign this cluster to be \(C_1\). Similarly, construct all the clusters to get the set \(C\). 
2. **Compute edges:** Given \(C\), use the Chow-Liu algorithm [14], which maps the edge estimation problem to finding a maximum weight spanning tree (MWST), to compute the edges \(E\).

To choose the clusters, we need to sort the correlations, that has complexity \(O(n \log n)\). If \(f(x)\) is Gaussian, Step 2 of Algorithm 1 has complexity \(O\left(mn^2 + \frac{mn^2}{m}\right)\), where the first term comes from computing the edges and the second term comes from the MWST. It is clear that the complexity of computing the edges dominates the complexity of choosing clusters.

3.2. Realizing RHT on Block-Tree Graphs
In this section, we show how the block-tree graph structure allows for recursive generation of samples, which is then used to compute the RHT. We will illustrate recursive structure for the case when \(f(x)\) is Gaussian with mean zero. For a block-tree graph \(G = (C, E)\), define \(C_1\) as the root cluster. The other clusters of the block-tree graph can be partially ordered according to their scale, which is the distance of a cluster to the root cluster \(C_1\). This distance is defined as the number of edges in the path connecting two clusters. Since we have a tree over the clusters, a path connecting two clusters is unique. Thus, the scale of \(C_1\) is zero. All the neighbors of \(C_1\) will have scale one. For any \(C_i\) at scale \(s\), define \(C_{\langle i\rangle}\) as the cluster connected to \(C_i\) at scale \(s - 1\). For example, in the block-tree graph in Fig. 1, let \(C_1 = \{1, 2, 3\}\) and \(C_2 = \{4, 5, 6\}\), then \(C_i = C_{\langle i\rangle}\). Following [7], a state-space model on the block-tree graph is

\[x_{C_i} = A_i x_{C_{\langle i\rangle}} + u_{C_i}, \]  
\[A_i = E(x_{C_i} x_{C_{\langle i\rangle}}^T) [E(x_{C_{\langle i\rangle}} x_{C_{\langle i\rangle}}^T)]^{-1} E(u_{C_i} u_{C_{\langle i\rangle}}^T) = E(x_{C_i} x_{C_{\langle i\rangle}}^T) - A_i E(x_{C_{\langle i\rangle}} x_{C_{\langle i\rangle}}^T) C_i, \]
(11)  
\(A_i \) is a white noise. To initiate the recursions in (11), we first find samples of \(x_{C_1}\) and then recursively generate the rest of the samples. To generate one sample from the block-tree graph, the time
complexity is $O(mn)$, since the main computation is that of matrix multiplication in (11). The algorithm to compute $c'$ and $c''$, and thus the RHT, is given as follows:

Algorithm 2: Computing $c'$ and $c''$.

1. For each hypothesis $f_k(x)$, compute block-tree graphs $G_k$ using a specified value of $m$. Using the aforementioned recursive sampling, generate sample sets $S_k$, $k = 0, 1$.
2. Using $S_0$ and $S_1$, computing $c'$ and $c''$ by solving (9) and (10) using Monte Carlo methods.

We remark that we are computing approximations of $c'$ and $c''$ since the random samples generated are not from $f_k(x)$, but instead from the density approximated using the block-tree graph. As noted earlier, generating samples from $f_k(x)$ is computationally intractable. Our algorithm for random sampling has similarities to blocked Gibbs sampling [15], where recursive generation of samples is achieved using the density $f(x_{C_i}, x_{C\setminus i})$. Computing this conditional density requires inversion of a large matrix rendering blocked Gibbs sampling to be computationally intractable. The block-tree graph framework finds the optimal cluster from $C_i$, to condition on, so as to find a balance between computational complexity and degree of approximation. Further, since the samples are generated from a sparse graphical model, the number of random samples that we need to generate are much less than what is needed had we sampled from $f(x)$.

3.3. Complexity Benefits

In this Section, we summarize the complexity benefits of using the block-tree graph framework for RHT. Assuming Gausianity, when generating samples directly from a distribution, the required time complexity is $O(n^3 + Ln^2)$. The first term comes from computing the singular value decomposition (SVD) of the covariance and the second term comes from generating $L$ samples. Using the block-tree graph framework with cluster size $m$, generating $L'$ samples has complexity $O(mn^2 + \frac{1}{m} \log \frac{1}{m} + Ln m)$. Since $m \ll n$, the block-tree graph only captures a small, but meaningful, number of statistical dependencies as compared to $f(x)$, so $L' \ll L$. Thus, we clearly see the complexity reductions when $n$ is large.

4. EXPERIMENTAL RESULTS: APPLICATIONS TO ROBUST CLASSIFICATION

4.1. RHT versus classical hypothesis testing

We first experimentally validate the claim that RHT is less sensitive to deviations from the assumed nominal probability densities, when compared to classical hypothesis testing. Let $F_k$ in (1) be chosen as mixtures of two multivariate Gaussians, such that the hypotheses correspond to $x \in \mathbb{R}^n$ coming from either nominal densities $f_0(x) \sim \mathcal{N}(\mu_0, \Sigma_0)$ or $f_1(x) \sim \mathcal{N}(\mu_1, \Sigma_1)$ with equal probability, and the corresponding contaminant densities being $h_0(x) \sim \mathcal{N}(0, \sigma_0^2 I)$ and $h_1(x) \sim \mathcal{N}(0, \sigma_1^2 I)$. We choose $n = 500$ and let $\epsilon_0 = \epsilon_1 = \epsilon$. The nominal densities are used to compute the classical Bayesian test and to generate samples for Monte Carlo simulations while performing RHT.

Figure 2 shows the variation in error probability with $\epsilon$. As expected, when $\epsilon = 0$, there is no deviation from the nominal density, and the error probability is the same for the classical hypothesis test as well as RHT. As $\epsilon$ increases, the benefits of RHT in terms of robustness to model choice are readily apparent.

4.2. Graph-based RHT versus RHT

As our primary investigation, we compare the performance of RHT vs. its realization on a graph which we will subsequently refer to as

"graph-based RHT". To reveal interesting trends, we work with empirical estimates of the nominal densities in Section 4.1, such that the these estimates are obtained for different training sample sizes. That is, training sets of varying sizes corresponding to each hypothesis are first generated using the actual nominal densities. The motivation for this experimental setup is in applications where the underlying densities are dynamic, however, training samples are only available at one time instant.

For RHT, the empirical estimates obtained from the training sets are used. Approximations to these empirical estimates via block-tree graphs (for a given $m$) are used for graph-based RHT. The two robust tests are then obtained by solving (9)-(10) for each case, yielding two sets of parameters, i.e. $(c', c'')_{RHT}$ and $(c', c'')_{graph-RHT}$. Figs. 3 and 4 report our findings, i.e. the classification error, as a function of training sample size. When the multi-variate nominal densities encode a fairly elaborate dependency structure between the random
variables (corresponding to a dense inverse of the covariance matrix), RHT has noticeable benefits over graph-based RHT. Interestingly, when the inverse of the covariance is sparse, i.e. the case of the density on the graph being a good approximation to the high-D nominal density, RHT and graph-based RHT are virtually indistinguishable. Further, the classification error is shown for different values of \( m = 10 \) and \( m = 20 \). Recall that when \( m \ll n \), graph-based RHT is orders of magnitude faster. It is remarkable therefore, that in Fig. 4 and for modest training in Fig. 3, the performance drop caused by using a graphical approximation is relatively small while yielding very significant computational benefits.

### 4.3. Real-world application: Target image classification

Finally, we apply robust hypothesis tests to a real-world classification problem occurring in automatic target recognition (ATR). Specifically, we perform binary classification on benchmark synthetic aperture radar (SAR) images acquired from the MSTAR database [16]. The images belong to one of two classes: T-72 tanks, and BMP-2 infantry fighting vehicles. The ATR problem is a suitable candidate for the following reasons: (i) features extracted from sensed imagery for target classification are invariably high-dimensional, (ii) access to training images is limited, and (iii) the sensed imagery is noisy, further leading to deviations from the true (unknown) class densities. The experimental set-up is briefly described next.

Feature vectors are extracted using the LL subband wavelet coefficients from a two-level wavelet decomposition of the SAR images. The feature vectors lie in \( \mathbb{R}^{256} \). Their true distributions \( f_0 \) and \( f_1 \) are unknown. Instead, we are given labeled training data \( T_0 \) and \( T_1 \) consisting of 232 vectors each. We therefore obtain empirical estimates \( f_0^* \) and \( f_1^* \) as a substitute. We model the densities corresponding to the two classes as multivariate Gaussian. Testing is performed on 1168 feature vectors picked from the two classes. We let \( m = 10 \).

We compare the performance of RHT using empirical distributions vs. graph-based RHT. Fig. 5 shows the variation in misclassification probability with the number of training samples for both cases. Three distinct regions can be identified: (i) region R1 in Figure 5 which corresponds to very low training and high error probability, (ii) region R2 with moderate (representative) training, and (iii) region R3 of large training. We cannot achieve true asymptotic training; instead we consider the set of all available features – training and test - as representative of asymptotic training. In region R2, the error using graph-based RHT is not significantly worse than that using RHT, albeit with a significant reduction in computational complexity. The misclassification error reduces as the number of training samples increases, since we obtain better approximations to the true distributions. The gap between errors widens as we approach the asymptotic case, but in reality, we are more likely to encounter situations which correspond to the region R2. Note that while we are while we are incurring a small loss of performance, using graphical models leads to significant run-time savings. In particular, in the region R2, the run-time for standard RHT was about 47 minutes using MATLAB on Intel Core2Duo 3.2 GB RAM CPU, while the graph-based RHT had an average run-time of about 4 minutes.

### 5. Conclusion

We considered the problem of designing minimax robust hypothesis tests (RHT) for high-dimensional signals. Traditional approaches to this problem are computationally intractable because of the need to solve a highly nonlinear equation using Monte Carlo methods. To enable efficient algorithms, we approximate distributions using graphical models, block-tree graphs in particular, in a way that allows for recursive algorithms to generate random samples. Experimental results on simulated and real world data sets reveal a favorable complexity vs. performance trade-off. Future work will involve studying more sophisticated algorithms that are still computationally efficient, to close the gap between the performance of graph-based RHT and true RHT. We will also investigate flavors of RHT (other than minimax) and their realization on graphs.

### 6. References