LEARNING MULTIPLE TARGET CONCEPTS FROM UNCERTAIN, AMBIGUOUS DATA USING THE ADAPTIVE COSINE ESTIMATOR AND SPECTRAL MATCH FILTER

By

JAMES M. BOCINSKY

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2019

For my wife, Kylie Bocinsky

ACKNOWLEDGMENTS

I would like to thank my adviser, Dr. Alina Zare, for all of her invaluable guidance, support, and encouragement during my graduate studies. Also I would like to thank the members of my thesis committee Dr. Paul Gader and Dr. Joe Wilson for their help and valuable suggestions.

Additionally, thank you to all of my lab mates and in particular Connor McCurley, Dylan Stewart, Matt Cook, and Daniel Shats for the valuable discussions, insight, and collaboration throughout my studies.

Thank you to my parents, John and Christy, and to my brother John for instilling a work ethic in me that has allowed me to complete my program. You have inspired me to be the best version of myself and I cannot thank you enough for always supporting me in my endeavors.

Lastly, I can not thank my wife, Kylie Bocinsky enough, for your love and support throughout my studies were invaluable. This would truly not be possible without your support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

LEARNING MULTIPLE TARGET CONCEPTS FROM UNCERTAIN, AMBIGUOUS DATA USING THE ADAPTIVE COSINE ESTIMATOR AND SPECTRAL MATCH FILTER

By

James M. Bocinsky

May 2019

Chair: Alina Zare
Major: Electrical and Computer Engineering

The Multiple Instance Adaptive Cosine Estimator and the Multiple Instance Subspace Match Filter are algorithms used in target detection, where a target class of interest is attempted to be detected amongst a non-target, background class. These algorithms learn a single feature vector representation to estimate a target class in a transformed feature space that normalizes the data to the background class. In this thesis, a number of algorithms are proposed to learn multiple target representations. These are evaluated using their respective performance and computation time using experiments containing a simulated hyperspectral dataset and the MUUFL Gulfport hyperspectral dataset captured over the campus of Southern Mississippi - Gulfport. The results are analyzed to conclude which variation of the multiple target algorithms is best in terms of performance and computation time.

CHAPTER 1
INTRODUCTION

In traditional supervised learning, each training data sample is paired with a corresponding label. However in many applications, obtaining data with sample level labels is often not possible. This can occur because it is too expensive, too time consuming, or the data presents itself in an uncertain, ambiguous manner. Uncertainty can present itself in data due to limitations of sensors. An example of this is using a Global Positioning System (GPS) to record ground truth information for an overhead, flight data collection. Due to the inaccuracies of the GPS, the collected data may exist in different pixel locations in the image than what is labeled by the GPS based ground truth. It is known that the pixel of interest is near the recorded ground truth, but it is uncertain if it is actually at that exact pixel location or not.

Furthermore, it can be infeasible to label individual instances due to cost or time limitations. It is more practical to label regions of data instead of individual instances, saving both the time and cost of processing data collections. For example, in image processing it is much easier to label an image as having an object of interest in it, rather than labeling each pixel where the exact object of interest exists. Additionally, if a bounding box is used to label where an object is in an image, there may exist other objects in the bounding box and bring about ambiguity in the data.

Learning from uncertain, ambiguous data has been an active area of research since the late 1990s and is known as multiple instance learning (MIL). This was formally constructed by Dietterich et al. (1997) when it was desired to learn which drugs are active in a mixture of drugs. In MIL, algorithms are designed to learn from multiple instances that are grouped together that share a label. These groups of data are known as bags. In many variations of MIL, like target detection or multiple instance regression, it is uncertain what the label of each instance is within each bag of data. These aspects make using traditional machine learning algorithms difficult, but by developing multiple instance learning algorithms, tasks that were once unfeasible before can be solved.

One application of multiple instance learning is performing target detection using hyperspectral data, where an algorithm attempts to learn representations of the training data to perform target detection against a background class. Hyperspectral sensors collect both spatial and spectral information by receiving reflected electromagnetic energy across a high number of wavelengths, often on the order of hundreds of wavelengths. Hyperspectral data is useful for classifying objects because the reflectance of various materials differ across different wavelengths of light making the received reflectance response unique for many different types of materials. Lastly, because the hyperspectral uses hundreds of bands, the data that is received is feature rich and allows for many different approaches to be explored in machine learning.

Even with these nice properties, hyperspectral data poses unique challenges. Firstly, the spatial resolution of hyperspectral sensors are often much lower than traditional digital cameras. This means that the objects of interest in an image can be at the subpixel level and need to be unmixed. Hyperspectral unmixing reduces down to two primary objectives. First, the reflectance of the item of interest needs to be estimated, known as endmember estimation. During this process, the spectral response corresponding to all items of interest are attempted to be extracted and estimated from the total spectral response. Secondly, the proportion of the spectral response for the item of interest is estimated, known as abundance estimation. This is often modeled as a convex mixing model which assumes each pixel is a combination of it's endmembers, shown in Equation (1-1) and (1-2).

$$\mathbf{x}_{\mathrm{i}} = \sum_{k=1}^{K} a_{ik}\mathbf{d}_{\mathrm{k}} + \varepsilon_i, i = 1, ..., N \tag{1-1}$$

$$\sum_{k=1}^{K} a_{ik} = 1, a_{ik} \geq 0, \forall i, k, \tag{1-2}$$

where N is the number of pixels, K is the number of endmembers or types of objects in the image, $\mathbf{x}_{\mathrm{i}}$ is a hyperspectral response for the i^{th} pixel, $\mathbf{d}_{\mathrm{k}}$ is the estimated endmember also

known as the spectral signature, and a_{ik} is the abundance for the k^{th} endmember at the i^{th} pixel.

Often within a hyperspectral image, the number of target instances is much fewer than the number of background instances. Within a hyperspectral image with a size of more than 100 x 100 pixels there may only be a few pixels that contain target. As mentioned previously, these target instances may even be subpixel as well. Due to this, it becomes very challenging to train traditional classifier models and algorithms have been developed to emphasis detection of target instances instead of overall classification.

Lastly, a global tracking system like a GPS is often used during hyperspectral data collections. As mentioned before, inaccurate GPS may cause the ground truth to be uncertain and the exact labels of pixels will not be accurate. This is commonly addressed in machine learning algorithms by using windows of data instead of single instances within an image. These problems naturally leads themselves to promote the use of a multiple instance learning algorithms for hyperspectral target detection.

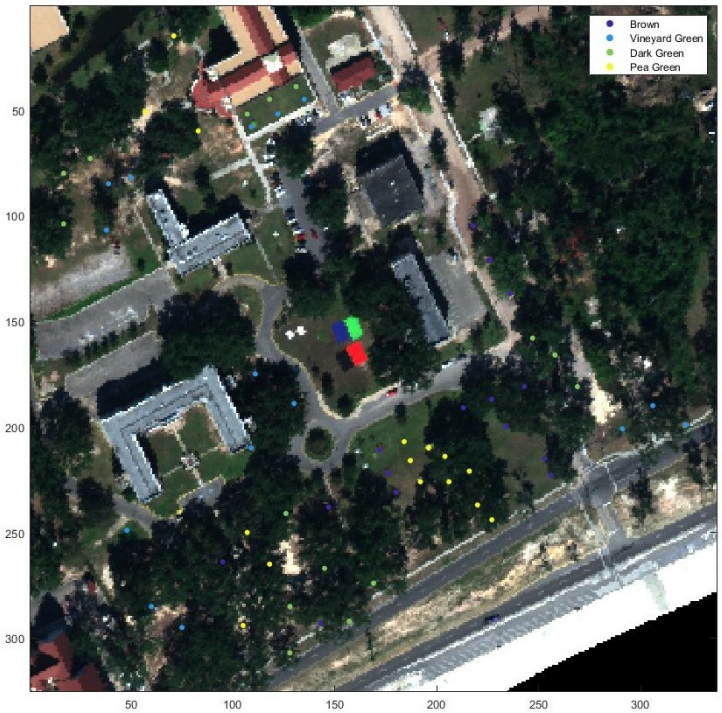


Figure 1-1. RGB Image of the MUUFL Gulfport dataset (Gader et al., 2013)

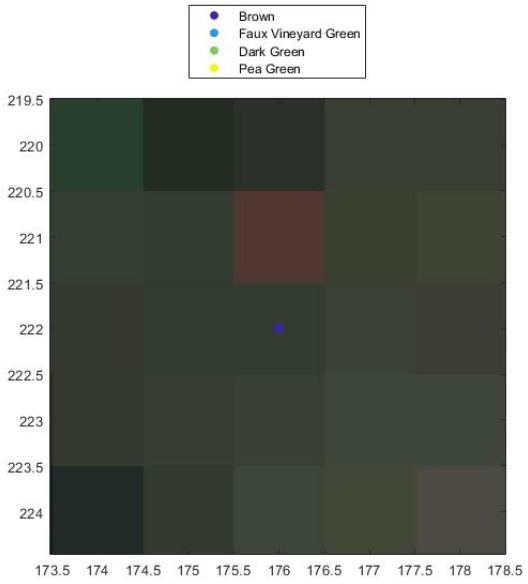


Figure 1-2. Zoomed in region of the RGB Image of the Gulfport dataset showing a brown target that is a pixel off of the recorded ground truth, showcasing the imprecise ground truth.

The primary objective of performing target detection using hyperspectral data is to detect unknown targets with a high rate of detection and a low false alarm rate. Other objectives like learning target concepts from the data or the proportions of the target concepts are of interest as well. A pair of multiple instance learning algorithms known as the multiple instance adaptive cosine estimator (MI-ACE) and the multiple instance spectral match filter (MI-SMF) have been used with hyperspectral data and have shown competitive results with other algorithms in terms of single target concept estimation and detection. A gap in research is if this pair of algorithms can be extended to learn multiple target concepts instead of just one. In the past, if these pair of algorithms were used to learn multiple target concepts, they were trained on each target type individually. Then, the results of each individual training would be combined to form a group of learned signatures, often referred to as a dictionary. Instead, the objective here is to develop an algorithm that will learn all of the target concepts at once without knowing

what target types exist in each of the positive bags. The research questions that are addressed in this thesis are listed below.

1. Between the variations of the multi-target algorithms proposed, is there a variation that is able to learn the underlying hyperspectral target signatures? If so, which multi-target method performs the best considering detection performance and computational efficiency?

2. Using a clustering initialization approach for the multi-target algorithms, which of the three clustering methods (K-Means, Ranked K-Means, and MI-ClusterRegress) performs the best, considering detection performance and computational efficiency?

3. Are any of the multi-target algorithm variations better, considering detection performance and computational efficiency, than other popular multiple instance learning algorithms such as the single target versions, the Adaptive Cosine Estimator (ACE) and Spectral Match Filter (SMF) detection statistics using manually selected target concepts, and the Multiple Instance Hybrid Estimator?

There is a gap in the literature that has not been explored, a multiple instance learning algorithm focusing on optimizing detection via the Adaptive Cosine Estimator (ACE) and the Spectral Match Filter (SMF), while learning multiple target concepts. This document will investigate three novel areas of initialization approaches and two novel optimization approaches to determine if any of these approaches can successfully learn multiple target concepts under a multiple instance learning framework. The objective of the proposed algorithms is to be able to learn multiple target concepts that are optimized for target detection using ACE or SMF, without knowing anything about the various target types that exist in uncertain labeled data.

CHAPTER 2
LITERATURE REVIEW

2.1 Multiple Instance Learning for Target Detection

Multiple instance learning is a subfield of machine learning where an algorithm or model learns from imprecise labels for various tasks including classification and regression. Multiple instance learning is often used for two class classification and target detection. Within multiple instance learning for target detection, individual feature vectors from a dataset, known as instances, are grouped into bags which are labeled as "positive" or "negative" based on their contents. A bag is labeled as positive if it contains at least one instance corresponding to a target class of interest, and a bag is labeled as negative if it contains only instances from the background, non-target class. A multiple instance algorithm for target detection will use this presumed knowledge to best estimate a target concept, in the form of a feature vector, also known as a target signature. The primary objective of multiple instance learning for target detection is to learn target signatures that best represent true positive instances while being as dissimilar to negative instances as possible.

2.1.1 Axis-parallel Rectangles

The Axis-parallel Rectangles (APR) techniques proposed by Dietterich et al. was used for determining the principal chemical compound drugs in a mixture of compounds that was responsible for making a mixture "active" Dietterich et al. (1997). This problem directly forms the multiple instance problem because the compound mixtures are only known to form an active bond or not. It is unknown which variation of the drug in the mixture is responsible for the activation. This application has been provided to show how a problem can fit into the MIL framework, but this algorithm is not limited to this application and has been used in many other applications. Putting this in terms of a general MIL framework, each mixture is considered a bag. Mixtures with an active drug are considered positive bags, and mixtures without an active drug are considered negative bags. The active drug is the feature vector of interest, also known as the target signature trying to be estimated.

The premise of the APR technique is to find a bounding box for every feature that includes an instance from each positive instance while excluding all of the instances from negative bags. An APR or bounding box is a set of thresholds, one for each feature dimension, that is used to discriminate between two classes. By using three different APR techniques shown in Figure 2-1, the authors were able to determine a target concept that was responsible for for the drug activation with differing levels of accuracy. The first APR technique, known as "GFS elim-count" (greedy feature selection elimination count), creates a bounding box that contains all positive instances, known as the "all-positive APR". Then the bounding box is iteratively shrunken to remove negative instances until all negative instances are removed. The second APR technique which is an outside-in approach called "GFS elim-kde" (greedy feature selection elimination kernel density estimation), also uses a bounding box with iterative shrinking to eliminate negative instances. This technique differs because it uses a cost function to estimate the cost of eliminating a positive instance, so that the negative instances that are associated with the least cost are eliminated first. Finally, the last APR technique which is an inside-out approach called "iterated-discrim" (iterated discrimination), takes an initial positive instance and grows a bounding box to contain at least one positive instance from every bag. These algorithms are explained in more detail in Sections 2.1.1.1 - 2.1.1.3. A relationship chart has been provided to show how the algorithms relate to each other in Figure 2-1.

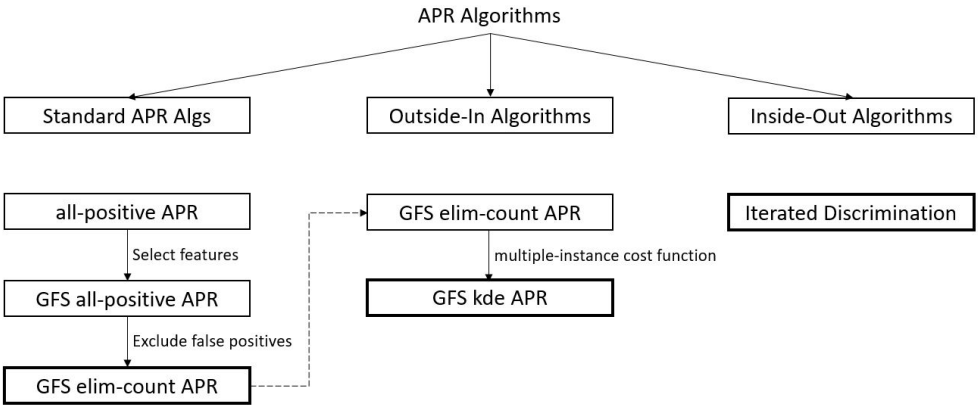


Figure 2-1. A visualization of how the 3 different APR techniques in bold boxes relate to each other.

2.1.1.1 GFS elim-count APR

The goal of the GFS elim-count algorithm is to create a bounding box that includes as many positive instances and as few negative instances as possible. It accomplishes this by first determining a bounding box APR that includes all instances from positive bags called the "all-positive APR". This initial APR is constructed without any regard to whether there are any negative instances inside the all-positive APR. As seen in Figure 2-2, the all-positive APR for 2 features, $x1$ and $x2$, is visualized as the bold, black box. Each different shape corresponds to a different bag. The instances from positive bags are the empty shapes, while the instances from negative bags are visualized as filled in shapes.

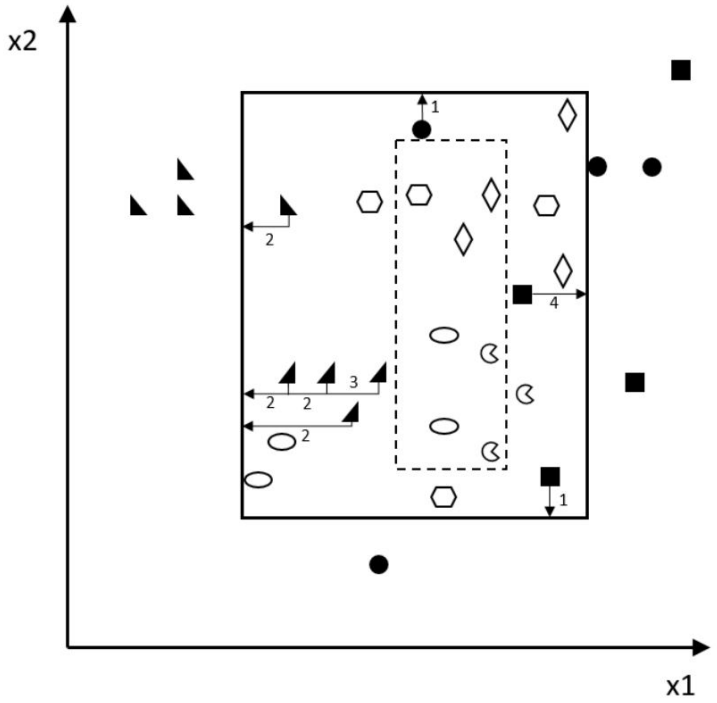


Figure 2-2. Representation of the GFS elim-count algorithm. Each shape is a different mixture where solid shapes are negative instances and empty shapes are instances from positive mixtures. The numbers on each negative instance represent the count, how many positive instances would be removed if the APR were to shrink on that "side."

After constructing the all-positive APR, the algorithm removes negative instances by shrinking the APR in an iterative manner. For each negative instance, it is determined the number of positive instances that would be removed if the APR were to shrink by removing that negative instance. These counts are then used to determine which "side" of the APR should be shrunk next. Here the side refers to one of the thresholds of the feature vector. This is done in an iterative manner until all negative instances have been removed from the APR. In Figure 2-2 you can see the result of GFS elim-count shown as the dashed box. The numbers next to each negative sample represent how many positive instances would be removed if that side was shrunk to remove that instance.

Although the GFS elim-count technique will determine an APR with only instances from positive bags, it does not address the multiple instance problem directly. The method does not take in to account if it will remove all of the instances from a particular positive bag, nor does it have any knowledge of what instance belongs to what bag. It merely knows whether how many positive instances will be removed if a particular negative instance is removed. With this, it can be seen that the algorithm will remove all of the instances from a positive bag if it is required to remove all of the negative instances from the APR.

2.1.1.2 GFS elim-kde APR

Dietterich et al. noticed the problem of GFS elim-count potentially removing all of the instances from a particular positive bag and wanted to address the problem directly. To address this they came up with a cost function for removing a positive instance. This cost function includes a Gaussian kernel density estimate (kde) to help estimate the likelihood of an instance being relevant. The cost of eliminating the i^{th} instance from bag j, denoted as $x_{j,i}$, depends on how alike it is to the other instances from bag j. This is modeled using the Gaussian kernel density estimate. A kernel is centered at each positive instance and estimates the likelihood of other positive instances being nearby in the feature space. If the instance has other nearby instances from the same bag, then it is given a high cost of elimination. Whereas if the instance is far away from other instances in the feature space, then it is an instance that should

not be included in the APR. Lastly, it can be seen in Equation (2-1) that if an instance is the last remaining instance in a positive bag, then it will have a very high cost of elimination.

$$\left(- \sum_{l=1, l \neq i}^{N_j} D_d(x_{j,l}) \right) + \alpha D_d(x_{j,i}) \tag{2-1}$$

In Equation (2-1), $D_d(x)$ is the kernel density estimate at point x along feature d, l is an index for all other positive instances remaining from bag j, N_j is the number of positive instances in bag j that remain in the APR, and α is a hyperparameter used to control how much an affect the second term plays. The first term corresponds to the cost of removing an instance from bag j, and the second term corresponds to how isolated the instance $x_{j,i}$ is along feature d. Breaking down this cost function there are three major takeaways.

1. The cost of removing data point $x_{j,i}$ is small if there are many other surviving positive instances from the same j^{th} positive bag.
2. The instance $x_{j,i}$ should be eliminated if other surviving positive instances from the same j^{th} bag have feature values that are frequently observed. This can be interpreted that those other survivors are the relevant instances that should be kept.
3. If the instance $x_{j,i}$ is isolated from other positive instances than it is not a relevant instance and should be removed.

This is an "outside in" approach which iteratively eliminates negative instances that have the least cost of elimination. Where the cost is the sum of the cost of removing the positive instances that would also be removed with the negative instance. When all of the negative instances have been removed the algorithm stops removing instances. At this point, the bounding box is determined and the algorithm is complete.

2.1.1.3 Iterated discrimination APR

Opposite of the "outside in" approach, the Iterated Discrimination APR takes an "inside out" approach. This approach starts with a single positive instance and grows an APR outward to include other positive instances.

The three steps of this algorithm are:

1. Grow - Grow an APR with tight bounds around the positive samples.

2. Discrim - Choose a set of discriminating features to use in the APR.

3. Expand - Expand the APR to be more general.

The first step is to grow an APR inside out to include other positive samples. The goal is to create the smallest APR that consists of at least one positive instance from every mixture. The size of an APR is defined in Equation (2-2).

$$Size(APR) = \sum_{d} ub_d - lb_d \tag{2-2}$$

where ub_d and lb_d are the upper and lower bound respectively of a dimension, d. The approach starts with an initial seed positive instance, and then greedily selects other positive instances to add in to it's APR. The positive instance added is the instance that would increase the size of the APR the least. On top of this, a backfitting algorithm is added. This algorithm still selects positive instances in a greedy manner, but it reconsiders all previous decisions every time a positive instance is added to the APR, known as backfitting. If an instance from the same mixture would decrease the size of the APR for that decision, the decision is changed and the instance is replaced with the new instance that makes the APR smaller.

The second step is to select discriminating features for the APR. A strong discriminating feature is one that discriminates against many negative instances and as well has a large distance in between the APR boundary and the closest negative instance. A strong feature is formally defined as one where:

1. the closest negative instance lies more than δ, a user set distance, outside the bounds of the APR along feature d

2. the closest negative instance lies the furthest away from the APR along feature d than any other feature

With this definition of a strong feature, the algorithm then iteratively selects features to select the bounds needed for the APR. The negative instances that were exterior to the APR from previous feature selections are not considered for further feature selection. The process of

selecting features to determine which bounds of the APR will be used is complete when all of the negative instances are external to the APR.

The iterated discrimination algorithm uses both the Grow and Discrim alternatively to construct the APR. First, an APR is constructed using all of the features. Then, a set of discriminating features are selected based on the original APR. Continuing, a new APR is constructed, but this time only using the discriminating features found from the previous APR. Then, another set of discriminating features are selected based on this APR. This process continues until it converges which typically takes 3-4 iterations.

Finally, the last step is to expand the APR to improve generalization. To accomplish this, the probability of a positive instance residing along a feature is used to extend the APR boundaries. To calculate the probability, a kernel density estimation of the positive instances is computed. By using the kernel density estimation, the probability of a positive instance being outside the expanded bounds is controlled by a hyperparameter, ε. The other hyperparameter that is selected is the width of the Gaussian kernel, τ, used for kernel density estimation. By expanding the APR's bounds, the model is able to improve generalization to other positive instances. This mitigates labeling positive instances that reside just outside the original bounds as negative instances.

2.1.2 Diverse Density

Diverse Density (DD) uses multiple instance learning to learn a target concept that is close to the intersection of many positive bags while being far from negative bags (Maron and Lozano-Pérez, 1998), (Maron and Ratan, 1998). Namely, DD uses the existing training data subspace, to learn a target concept. The target concept comes from a region with a high "density" positive instances from different bags that is far from negative instances, thus the name Diverse Density. By using the existing data space and choosing the region with a maximum diverse density, the algorithm can select a representation that learns the true concept of a target.

A probabilistic measure to estimate diverse density was proposed to determine what region of the manifold has the most amount of instances that came from different positive bags. This can be seen in Equation (2-3), where $\mathbf{x}$ are instances from the j^{th} bag, $\mathbf{t}$ is a true target concept that is being learned, $\mathbf{B}_j^+$ and $\mathbf{B}_j^-$ are the j^{th} positive and negative bags respectively.

$$\arg\max_{x} \prod_j \mathbf{Pr}(\mathbf{x} = \mathbf{t}|\mathbf{B}_j^+) \prod_j \mathbf{Pr}(\mathbf{x} = \mathbf{t}|\mathbf{B}_j^-) \tag{2-3}$$

To calculate the terms in the general diverse density measure, Equation (2-3), a noisy-or model was assumed (Srinivas, 1993). This model shows that the joint probabilities can be calculated as shown in Equation (2-4) and (2-5).

$$\mathbf{Pr}(\mathbf{x} = \mathbf{t}|\mathbf{B}_j^+) = \mathbf{Pr}(\mathbf{x} = \mathbf{t}|\mathbf{B}_{j1}^+, \mathbf{B}_{j2}^+, ..., \mathbf{B}_{jN_j}^+) = 1 - \prod_i (1 - \mathbf{Pr}(\mathbf{x} = \mathbf{t}|\mathbf{x}_{ij} \in \mathbf{B}_j^+)) \tag{2-4}$$

$$\mathbf{Pr}(\mathbf{x} = \mathbf{t}|\mathbf{B}_j^-) = \prod_j (1 - \mathbf{Pr}(\mathbf{x} = \mathbf{t}|\mathbf{x}_{ij} \in \mathbf{B}_j^-)) \tag{2-5}$$

The probability of an individual instance being a potential target is shown in Equation (2-6).

$$\mathbf{Pr}(\mathbf{x} = \mathbf{t}|\mathbf{x}_{ij}) = \exp(-\|\mathbf{x}_{ij} - \mathbf{x}\|)^2 \tag{2-6}$$

This probability measure is proportional to a Gaussian distribution and measures the distance between an instance, $\mathbf{x}_{ij}$, and the potential target, $\mathbf{x}$. If one of the instances from a positive bag is close to $\mathbf{x}$, then the probability is high. More so, if all of the positive bags have one instance that is close to $\mathbf{x}$, and no negative bags are close to $\mathbf{x}$, then $\mathbf{x}$ will have a even higher diverse density. It can also be seen that any additional instance from a bag that is not yet near $\mathbf{x}$ will give an exponential increase to the diverse density. Similarly, if any negative instance is near, it will drive down the diverse density.

Lastly, DD not only determines a target concept, but also determines weights for the target's features to determine which features are relevant. To accomplish this, the distance between two points $\mathbf{x}_{ij}$ and $\mathbf{x}$ in the feature space is modified to include a weight as shown in Equation (2-7).

$$\|\mathbf{x}_{ij} - \mathbf{x}\|^2 = \sum_k w_k (\mathbf{x}_{ijk} - \mathbf{x}_k)^2 \tag{2-7}$$

Where each k^{th} feature will have a learned weight associated to it. To learn the weights a gradient ascent algorithm is used. Since the data space can be large in some applications, only the positive instances are considered as starting points for gradient ascent. The location in the manifold space that provides the largest diverse density measure is the region where the learned target concept exists.

2.1.3 Expectation Maximization Diverse Density

Expectation Maximization Diverse Density (EM-DD) is an extension of the DD algorithm where EM is used to select which datapoint in each bag is most likely responsible for providing the label for that bag (Zhang and Goldman, 2002). For example, EM is used to determine which instance from a positive bag is most likely the target representative in that bag that gives that bag's label a positive label. As stated by Zhang and Goldman (2002) when using the Musk data set, the EM-DD algorithm is much more efficient, by 10-100 times, because it operates on only one instance from each bag. Namely, the DD algorithm is still used but operates on the data points from each positive bag that is most likely the target from that bag. By doing this the algorithm is more efficient because it only needs to consider these data points when calculating the Diverse Density metric.

To operate on the instance level of data, EM is used to determine which instance from a bag is responsible for that bag's label, a missing attribute, and works as follows. During the expectation step (E-step), the datapoint from each positive bag that is most like the current estimation of a target concept, $\mathbf{t}$, is determined. This is known as the bag representative and is selected using the author's generative model, shown in Equation (2-8).

$$\mathbf{x}_j^* = \arg\max_{\mathbf{x}_{ij} \in \mathbf{B}_j} \exp(-\|\mathbf{x}_{ij} - \mathbf{t}\|)^2 \tag{2-8}$$

During the maximization step (M-step), a new target concept, $\mathbf{t}'$ is estimated. This is accomplished by using all of the bag representatives from the E-step, calculating their diverse density metric, and using gradient ascent on the bag representative that maximizes the diverse density metric to estimate a new target concept, shown in Equation (2-9). Then, the algorithm returns back to the E-step and repeats this process until the algorithm converges or the maximum number of iterations is reached.

$$\mathbf{t}' = \arg\max_{\mathbf{t}} \prod_j \mathbf{Pr}(L_j|\mathbf{t}, \mathbf{x}_j^*) \tag{2-9}$$

2.1.4 Functions of Multiple Instances

The Functions of Multiple Instances (FUMI) algorithm extends the approach of the Sparsity Promoting Iterated Constrained Endmember (SPICE) algorithm (Zare and Gader, 2007). The SPICE algorithm is an unsupervised algorithm that learns the proportions and endmembers of an unlabeled dataset. An endmember is a pure response of a object of interest. FUMI extends the SPICE algorithm by using labeled data to learn a target endmember (or prototype) as well as non-target prototypes (Zare and Gader, 2010). By learning these prototypes, FUMI is able to use the prototypes to perform target detection on a dataset with unknown labels. Since the origination of FUMI, different variations of the algorithm have been created. In this literature review the original FUMI algorithm, C-FUMI, short for convex FUMI, and an extension of this, E-FUMI are discussed.

2.1.4.1 Convex FUMI

The objective of Convex FUMI (C-FUMI) is to learn a target prototype, several non-target prototypes, and the number of non-target prototypes needed. To accomplish this, each binary labeled data point of the training data will be estimated as a proportion of learned prototypes. In this algorithm a set of prototypes, $\mathbf{E}$, is learned, where $\mathbf{e}_\mathrm{T}$ is the learned target prototype,

and $\mathbf{e}_j$ and $\mathbf{e}_k$ are the j^{th} and k^{th} non-target prototype. Proportions of each data point are also estimated using a weight p_{ik} for the k^{th} prototype in data point i. This is represented by Equation (2-10).

$$\mathbf{x}_i = p_{iT}\mathbf{e}_T + \sum_{k=1}^{M} p_{ik}\mathbf{e}_k \tag{2-10}$$

The data points labeled as target must have some unknown positive weight p_{iT} for the target prototype e_T, i.e., $\mathbf{x}_i^+ = p_{iT}\mathbf{e}_T - \sum_{k=1}^{M} p_{ik}\mathbf{e}_k$ where $p_{iT} > 0$. Furthermore, the non-target labeled data points must have zero weight for the target prototype, i.e., $\mathbf{x}_i^- = \sum_{k=1}^{M} p_{ik}\mathbf{e}_k$.

In SPICE, the algorithm that FUMI extends, the prototypes and proportions are updated by minimizing the objective function shown in Equation (2-11). SPICE is an unsupervised algorithm that uses alternating optimization on the learned endmembers and proportions to minimize the objective function.

The objective function can be broken down into three terms. The first term computes the squared error between data point $\mathbf{x}_i$ and it's proportions of prototypes (*endmembers*). The second term minimizes the distance between prototypes providing a tight fit of prototypes around the data. The third term promotes sparsity which allows the algorithm to determine M, the number of non-target prototypes needed, where $\gamma_k = \frac{\Gamma}{\sum_{i=1}^{N} p_{ik}}$. Γ is a parameter used to control the level of sparsity of the learned prototypes.

$$G = (1-\mu)\sum_{i=1}^{N}\left\|\left(\mathbf{x}_i - \sum_{k=1}^{M} p_{ik}\mathbf{e}_k\right)\right\|_2^2 + \frac{\mu}{2}\sum_{k=1}^{M}\sum_{j=1}^{M}\|(\mathbf{e}_k - \mathbf{e}_j)\|_2^2 + \sum_{k=1}^{M}\gamma_k\sum_{i=1}^{N} p_{ik} = (1-\mu)R + \frac{\mu}{2}V + S \tag{2-11}$$

$$F = (1-\mu)\sum_{i=1}^{N}\left\|\left(\mathbf{x}_i - I(\mathbf{x}_i)p_{iT}\mathbf{e}_T - \sum_{k=1}^{M} p_{ik}\mathbf{e}_k\right)\right\|_2^2 + \frac{\mu}{2}R + \mu\sum_{k=1}^{M}\|(\mathbf{e}_T - \mathbf{e}_k)\|_2^2 + S + \sum_{\substack{i=1 \\ I(\mathbf{x}_i)=1}}^{N^+}\frac{1}{\sigma^2}(p_{iT}-1)^2 \tag{2-12}$$

C-FUMI extends SPICE by using data that is labeled as target or non-target, to learn a target prototype and multiple non-target prototypes. The objective function is similar to SPICE and is shown in Equation (2-12). Some of the terms are different. One key difference is found in the first term. l is 1 when $\mathbf{x}_i$ is from the target class, and 0 when $\mathbf{x}_i$ is from the non-target class. This ensures the constraint that non-target instances can be represented by using only the non-target prototypes. Again, the fourth term helps the algorithm learn the number of non-target prototypes. When the proportions of an unneeded prototype are near zero, that prototype is removed without affecting the squared error terms. The fifth term uses a hyperparameter, σ^2, which can be adjusted based on the uncertainty of the labeling, or prior knowledge about the proportions of target endmembers in the positive bags. σ^2 should be large if there are a small proportion of target endmembers in the positive bags.

After the prototypes have been learned, Zare and Gader (2010) proposed target detection can be done in two different manners. The first way to perform target detection is by computing the residual sum of squared errors of a test sample and the proportions of the learned prototypes. The proportion value of the target concept is used as the detection statistic. This is shown in Equation (2-13).

$$\left\| \left(\mathbf{x}_i - p_{iT}\mathbf{e}_T - \sum_{k=1}^{M} p_{ik}\mathbf{e}_k \right) \right\|_2^2 \tag{2-13}$$

The second method proposed for using the prototypes for target detection is one that is inspired by the Hybrid Subpixel Detector method (Broadwater and Chellappa, 2007). The detection statistic is the ratio of the residual errors from both sets of prototypes, where the numerator corresponds to the residual error of the test sample being represented as non-target prototypes, and the denominator is the residual error of the test sample being represented as the target and non-target prototypes. This is shown in Equation (2-14).

$$\frac{\left\| \left(\mathbf{x}_\mathrm{i} - \sum_{k=1}^{M} p_{ik}^{*} \mathbf{e}_\mathrm{k} \right) \right\|_2^2}{\left\| \left(\mathbf{x}_\mathrm{i} - p_{iT} \mathbf{e}_\mathrm{T} - \sum_{k=1}^{M} p_{ik} \mathbf{e}_\mathrm{k} \right) \right\|_2^2} \tag{2-14}$$

2.1.4.2 Extended FUMI

An extended version of C-FUMI, called E-FUMI, allows for bag level training label uncertainty (Jiao and Zare, 2015). In the context of images, E-FUMI only requires the data to labeled as an approximation of where a target prototype may exist. In other words, the labels only specify that at least one instance within a region must contain some proportion of the target prototype, but it is unknown which instances within that region contain the target prototype. In this regard, the E-FUMI algorithm fits the Multiple Instance Learning (MIL) framework more clearly, where a region with a proportion of target prototype would be considered a positive bag, and a region without any target prototypes would be considered a negative bag.

To solve this problem, an Expectation Maximization (EM) approach is used. The hidden latent variables for this problem are the instance-level labels. The complete data log-likelihood for this problem is shown in Equation (2-15). This is an extension of C-FUMI and it can be seen that the labels for each instance in C-FUMI, $l(\mathbf{x}_\mathrm{i})$, are replaced with the latent variables z_i.

$$\begin{aligned} F = & \frac{(1-\mu)}{2} \sum_{i=1}^{N} w_i \left\| \left(\mathbf{x}_\mathrm{i} - z_i p_{iT} \mathbf{e}_\mathrm{T} - \sum_{k=1}^{M} p_{ik} \mathbf{e}_\mathrm{k} \right) \right\|_2^2 \\ & + \frac{\mu}{2} \sum_{k=1}^{M} \|\mathbf{e}_\mathrm{k} - \mu_0\|_2^2 + \frac{\mu}{2} \|\mathbf{e}_\mathrm{T} - \mu_0\|_2^2 + \sum_{k=1}^{M} \gamma_k \sum_{i=1}^{N} p_{ik} \end{aligned} \tag{2-15}$$

$$E[F] = \sum_{z_i \in \{0,1\}} \left[\frac{(1-\mu)}{2} \sum_{i=1}^{N} w_i P\left(z_i | \mathbf{x}_\mathrm{i}, \boldsymbol{\Theta}^{(\mathrm{t}-1)}\right) \left\| \mathbf{x}_\mathrm{i} - z_i p_{iT} \mathbf{e}_\mathrm{T} - \sum_{k=1}^{M} p_{ik} \mathbf{e}_\mathrm{k} \right\|_2^2 \right]$$
$$+ \frac{\mu}{2} \sum_{k=1}^{M} \|\mathbf{e}_\mathrm{k} - \mu_0\|_2^2 + \frac{\mu}{2} \|\mathbf{e}_\mathrm{T} - \mu_0\|_2^2 + \sum_{k=1}^{M} \gamma_k \sum_{i=1}^{N} p_{ik} \tag{2-16}$$

The latent variables for each data point, z_i, are unknown and must be estimated using Equation (2-16) during the Expectation step (E-step) of the algorithm. In Equation (2-16), $\Theta^{(t)}$ is the set of parameters at the current iteration t. $P(z_i|\mathbf{x}_\mathrm{i}, \boldsymbol{\Theta}^{(\mathrm{t}-1)})$ is the probability of individual points having any proportion of target or non-target in them. The other terms of Equation (2-16) are the same as from the objective function in Section 2.1.4.1. The $P(z_i|\mathbf{x}_\mathrm{i}, \boldsymbol{\Theta}^{(\mathrm{t}-1)})$ is determined using the previous iterations parameters and the constraints which follow in Equation (2-17).

$$P(z_i|\mathbf{x}_\mathrm{i}, \boldsymbol{\Theta}^{(\mathrm{t}-1)}) = \begin{cases} e^{-\beta r_b} & \text{if} \quad z_i = 0, L_i = 1 \\ 1 - e^{-\beta r_b} & \text{if} \quad z_i = 1, L_i = 1 \\ 0 & \text{if} \quad z_i = 1, L_i = 0 \\ 1 & \text{if} \quad z_i = 0, L_i = 0 \end{cases} \tag{2-17}$$

where β is a scaling parameter and $r_b = \left\| \mathbf{x}_\mathrm{i} - \sum_{k=1}^{M} p_{ik} \mathbf{e}_\mathrm{k} \right\|_2^2$. r_b represents the residual error between a datapoint and the background prototypes. By analyzing the constraints we can see that when a data point from a positive bag, $L_i = 1$ is truly a non-target, it should have a low residual error, causing the $P(z_i = 0|\mathbf{x}_\mathrm{i}, \boldsymbol{\Theta}^{(\mathrm{t}-1)}) = e^{-\beta r_b} \rightarrow 1$. Furthermore, when the data point is truly a target, the residual error should be high and $P(z_i = 1|\mathbf{x}_\mathrm{i}, \boldsymbol{\Theta}^{(\mathrm{t}-1)}) = 1 - e^{-\beta r_b} \rightarrow 1$. When instances from a negative bag are considered, $L_i = 0$, it is known that the instances must be truly non-target and the probabilities are set to their respective 0 or 1 value. With this in place, the E-step can be computed.

The Maximization step (M-step) is done by optimizing Equation (2-16) for the unknown parameters given the E-step's current expectation values of the latent variables. The parameters that are optimized are the prototypes, $\mathbf{e}_k$, and their proportion values, p_{ik}. Lastly, in the M-step, if their are similar prototypes, these prototypes are pruned to reduce the number of prototypes needed to represent the data.

By using the EM algorithm, E-FUMI is able to estimate the instance-level labels and work within a MIL framework. This allows for a greater uncertainty of your data and only requires a region of data to be specified as having an existing target, not necessarily which points within that region contain a target or not.

2.1.5 Multiple Instance Adaptive Cosine Estimator/Spectral Match Filter

The Multiple Instance Adaptive Cosine Estimator (MI-ACE) and Multiple Instance Spectral Match Filter (MI-SMF) were originally proposed by Zare et al. (2018) to solve common problems with performing target detection using detection metrics for various applications. The two detection metrics that these algorithms make use of are the Adaptive Cosine Estimator (ACE) and the Spectral Match Filter (SMF). To use these detection metrics a target signature must be known prior to performing detection. Techniques to estimate target representatives can be measured in a laboratory setting, but are often unrealistic and not representative of a target in various conditions and environments. Alternatively, a target representation may be extracted directly from the data itself. Often times when this is done, the extracted representation does not contain meaningful features to differentiate it from the background and may not provide the desired performance. This is because the extracted representation is in it's original data subspace, which may not provide relevant features to discriminate it from the background. To handle this MI-ACE and MI-SMF apply a data transformation known as whitening to transform the data to have zero mean and unit variance with respect to the background data. This transforms the data so that the remaining features have equal weight and should not contain features that come from the background of the data.

Lastly, it is often times difficult or even impossible to extract a target representation from a dataset. For example with hyperspectral data, the target signature is often times at the subpixel level so it is impossible to extract the exact desired representation from the data. Furthermore, in the context of explosive hazard detection, the exact boundaries of an explosive hazard's response within a physical sweep is impossible to obtain, and thus determining where to extract a target representation is nonviable. The MI-ACE and MI-SMF algorithms address these problems and are able to learn a target signature that is optimal for the ACE and SMF detection metrics respectively. By using MI-ACE and MI-SMF, a more true representation of a target signature can be learned.

MI-ACE and MI-SMF follow the multiple instance learning framework where the labels of the data are at the bag level. With this, let $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ be a training dataset with each sample, $\mathbf{x}_i$ being a vector with dimensionality D. The data is grouped into J bags $\mathbf{B} = \{\mathbf{B}_1, ..., \mathbf{B}_J\}$ with labels, $L = \{L_1, ..., L_J\}$, where $L_j \in \{0, 1\}$.

A bag is considered positive, $\mathbf{B}_j^+$, with label, $L_j = 1$, when there exists at least one instance, $\mathbf{x}_{ji}$, in bag j that is from the target class, $l_{ji} = 1$, seen in Equation (2-18). Additionally, a bag is considered negative, $\mathbf{B}_j^-$, with label $L_j = 0$, if all instances in bag j are from the background class, $l_{ji} = 0$, seen in Equation (2-19). The number of instances in both positive and negative bags is variable.

$$\text{if} \quad L_j = 1, \quad \exists \mathbf{x}_{ji} \in \mathbf{B}_j^+ \quad \text{s.t.} \quad l_{ji} = 1 \tag{2-18}$$

$$\text{if} \quad L_j = 0, \quad \forall \mathbf{x}_{ji} \in \mathbf{B}_j^-, \quad l_{ji} = 0 \tag{2-19}$$

With this formulation, the goal of MI-ACE and MI-SMF is to estimate a target signature, $\mathbf{s}$, that maximizes the detection statistic of the target instances in the positive bags while minimizing the detection statistic of all negative instances. This can be accomplished by

maximizing the objective shown in Equation (2-20),

$$\arg\max_{s} \frac{1}{N^+} \sum_{j:L_j=1} D(\mathbf{x}_j^*, \mathbf{s}) - \frac{1}{N^-} \sum_{j:L_j=0} \frac{1}{N_j^-} \sum_{x_i \in B_j^-} D(\mathbf{x}_i, \mathbf{s}) \tag{2-20}$$

where N^+ is the number of positive bags, N^- is the number of negative bags, and N_j^- is the number of instances in negative bag j. $\mathbf{x}_j^*$ is the positive instance selected from bag j that is most like the target signature, $\mathbf{s}$. This is known as the bag representative and is shown in Equation (2-21).

$$\mathbf{x}_j^* = \arg\max_{x_i \in B_j^+} D(\mathbf{x}_i, \mathbf{s}) \tag{2-21}$$

The two detection statistics are SMF and ACE. SMF, shown in Equation (2-22), is the projection of a test sample, $\mathbf{x}$, onto a known target signature, $\mathbf{s}$, in a whitened coordinate space (Kraut and Scharf, 1999), (Kraut et al., 2001), and (Nasrabadi, 2008). The whitening is done using the background covariance, $\mathbf{\Sigma}_b^{-1}$, and background mean, $\boldsymbol{\mu}_b$, to transform the background to have zero mean and a uniform, unit variance. Finally, it can be seen that the statistic is normalized by the target signature in the whitened coordinate space. With this, not only does the projection of the test sample onto the target signature affect the statistic, but the magnitude of the test sample will additionally affect the statistic.

$$D_{SMF}(\mathbf{x}, \mathbf{s}) = \frac{\mathbf{s}^T \mathbf{\Sigma}_b^{-1} (\mathbf{x} - \boldsymbol{\mu}_b)}{\sqrt{\mathbf{s}^T \mathbf{\Sigma}_b^{-1} \mathbf{s}}} \tag{2-22}$$

ACE shown in Equation (2-23), is also the projection of a test sample, $\mathbf{x}$, onto a known target signature, $\mathbf{s}$, in a whitened coordinate space (Kraut and Scharf, 1999), (Kraut et al., 2001), and (Basener, 2010). Again, the whitening is done using the background covariance, $\mathbf{\Sigma}_b^{-1}$, and background mean, $\boldsymbol{\mu}_b$, to transform the background to have zero mean and a uniform, unit variance. The difference here is that ACE is normalized by not only the target signature, $\mathbf{s}$, but the whitened test sample, $\mathbf{x}$, as well. With this, contrary to SMF, the

magnitude of the test sample will not affect the statistic, and only the shape of the feature vector contributes to the statistic.

$$D_{ACE}(\mathbf{x},\mathbf{s}) = \frac{\mathbf{s}^{\top}\boldsymbol{\Sigma}_{\mathrm{b}}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{\mathrm{b}})}{\sqrt{\mathbf{s}^{\top}\boldsymbol{\Sigma}_{\mathrm{b}}^{-1}\mathbf{s}}\sqrt{(\mathbf{x}-\boldsymbol{\mu}_{\mathrm{b}})^{\top}\boldsymbol{\Sigma}_{\mathrm{b}}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{\mathrm{b}})}} \tag{2-23}$$

These detection statistics can be rewritten to allow for better notation. This is shown in the following:

$$D_{ACE}(\mathbf{x},\mathbf{s}) = \frac{\mathbf{s}^{\top}\boldsymbol{\Sigma}_{\mathrm{b}}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{\mathrm{b}})}{\sqrt{\mathbf{s}^{\top}\boldsymbol{\Sigma}_{\mathrm{b}}^{-1}\mathbf{s}}\sqrt{(\mathbf{x}-\boldsymbol{\mu}_{\mathrm{b}})^{\top}\boldsymbol{\Sigma}_{\mathrm{b}}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{\mathrm{b}})}} \tag{2-24}$$

$$= \frac{\mathbf{s}^{T}\mathbf{U}\mathbf{D}^{-\frac{1}{2}}\mathbf{D}^{-\frac{1}{2}}\mathbf{U}^{T}(\mathbf{x}-\boldsymbol{\mu}_{\mathrm{b}})}{\sqrt{\mathbf{s}^{T}\mathbf{U}\mathbf{D}^{-\frac{1}{2}}\mathbf{D}^{-\frac{1}{2}}\mathbf{U}^{T}s}\sqrt{(\mathbf{x}-\boldsymbol{\mu}_{\mathrm{b}})^{\top}\mathbf{s}^{T}\mathbf{U}\mathbf{D}^{-\frac{1}{2}}\mathbf{D}^{-\frac{1}{2}}\mathbf{U}^{T}(\mathbf{x}-\boldsymbol{\mu}_{\mathrm{b}})}} \tag{2-25}$$

$$= \left(\frac{\hat{\mathbf{s}}}{||\hat{\mathbf{s}}||}\right)^{T}\left(\frac{\hat{\mathbf{x}}}{||\hat{\mathbf{x}}||}\right) \tag{2-26}$$

$$= \hat{\hat{\mathbf{s}}}^{T}\hat{\hat{\mathbf{x}}} \tag{2-27}$$

where $\hat{\mathbf{x}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{U}^{T}(\mathbf{x}-\boldsymbol{\mu}_{\mathrm{b}})$, $\hat{\mathbf{s}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{U}^{T}\mathbf{s}$, $\mathbf{U}$ and $\mathbf{D}$ are the eignenvectors and eigenvalues of the background covariance, $\boldsymbol{\Sigma}_{\mathrm{b}}^{-1}$, respectively, $\hat{\hat{\mathbf{s}}} = \frac{\hat{\mathbf{s}}}{||\hat{\mathbf{s}}||}$, and $\hat{\hat{\mathbf{x}}} = \frac{\hat{\mathbf{x}}}{||\hat{\mathbf{x}}||}$. In this case, the notation for ACE has been rewritten, but the same steps can be applied to SMF and rewritten except that $\hat{\mathbf{x}}$ would be used instead of $\hat{\hat{\mathbf{x}}}$ in the SMF version of Equation (2-27).

To estimate the target signature, the objective function in Equation (2-20) is maximized. To accomplish this, the algorithm is broken up in to two primary steps, initializing a target signature, and then optimizing that signature using one target from each positive bag, also known as the bag representative, $\mathbf{x}_{\mathrm{j}}^{*}$. To initialize the target signature, the objective function is computed for all of the positive instances and which ever instance provides the largest objective function becomes the initialized target signature. Although this instance may provide the highest objective function value, it may not be optimal for all of the positive instances

within the data. So considering this, optimization is done using the update equation shown in Equation (2-28). This is derived from the associated Lagrangian problem.

$$\hat{\hat{\mathbf{s}}} = \frac{\mathbf{t}}{||\mathbf{t}||} \quad \text{where} \quad \mathbf{t} = \frac{1}{N^+} \sum_{j:L_j=1} \hat{\hat{\mathbf{x}}}_j^* - \frac{1}{N^-} \sum_{j:L_j=0} \frac{1}{N_j^-} \sum_{\mathrm{x_i} \in B_j^-} \hat{\hat{\mathbf{x}}}_i \tag{2-28}$$

To optimize the initialized signature, the signature, $\hat{\hat{\mathbf{s}}}$, is iteratively updated using Equation (2-28). In each iteration, the current bag representatives, $\mathbf{x}_j^*$, are determined given the current estimated target signature. The bag representatives are averaged and then the average of the background samples is subtracted away. Finally the target signature is normalized and the updated target signature has been computed. The average background will not change from iteration to iteration so this term can be precomputed.

The algorithm pseudocode is provided below:

Algorithm 1 MI-SMF/MI-ACE

1: Compute $\boldsymbol{\mu}_\mathrm{b}$ and $\boldsymbol{\Sigma}_\mathrm{b}$
2: Subtract the background mean and whiten all instances, $\hat{\mathbf{x}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{U}^T(\mathbf{x} - \boldsymbol{\mu}_\mathrm{b})$
3: **if** MI-ACE **then**
4: normalize: $\hat{\hat{\mathbf{x}}} = \frac{\hat{\mathbf{x}}}{||\hat{\mathbf{x}}||}$
5: **end if**
6: Initialize $\hat{\hat{\mathbf{s}}}$, with the instance in a positive bag that maximizes the obj. fun., Equation (2-20)
7: **repeat**
8: Update the bag representatives, $\mathbf{x}_j^*$, for each positive bag, $\mathbf{B}_j^+$, using Equation (2-21)
9: Update $\hat{\hat{\mathbf{s}}}$ using Equation (2-28)
10: **until** Stopping Criterion Reached
11: **return** Normalized De-whitened target signature, $\mathbf{s} = \frac{\mathbf{t}}{||\mathbf{t}||}$, where $\mathbf{t} = \mathbf{U}\mathbf{D}^{-\frac{1}{2}}\hat{\hat{\mathbf{s}}}$

2.1.6 Multiple Instance Hybrid Estimator

The Multiple Instance Hybrid Estimator (MI-HE) algorithm learns multiple target signatures and multiple background signatures, making it the most similar to the proposed algorithms in this thesis. The MI-HE algorithm aims to maximize the objective function in Equation (2-29), (Jiao and Zare, 2017), (Jiao et al., 2018). Namely, MI-HE wants to maximize the probability that positive bags are labeled positive and negative bags are labeled as negative.

Since MI-HE works within the multiple instance framework, the objective function can be simplified to only need to maximize a single instance from each positive bag. Furthermore, within the multiple instance framework, every instance from a negative bag is truly a negative instance. So all negative instances should be labeled as negative. These two modifications of the objective function are represented in Equation (2-30).

$$J_1 = \prod_{j=1}^{N^+} \Pr(L_j^+ = +|\mathbf{B}_j^+) \prod_{j=1}^{N^-} \Pr(L_j^- = -|\mathbf{B}_j^-) \tag{2-29}$$

$$= \prod_{j=1}^{N^+} \max_{i \in N_j^+} \Pr(I_{ij}^+ = +|\mathbf{B}_j^+) \prod_{j=1}^{N^-} \prod_{i=1}^{N_j^-} \Pr(I_{ij}^- = -|\mathbf{x}_{ij}^-) \tag{2-30}$$

The max operation was adapted by many models to fall into the noisy-OR model (Srinivas, 1993) that is commonly used in multiple instance learning. The authors modified this to implement a generalized mean instead of using the max operation. This modification is shown in Equation (2-31), where the hyperparameter $p \in [-\infty, \infty]$ varies the operation from a min to a max, respectively.

$$J_2 = \prod_{j=1}^{N^+} \left(\frac{1}{N_j^+} \sum_{i=1}^{N_j^+} \Pr(I_{ij}^+ = +|\mathbf{B}_j^+)^p \right)^{\frac{1}{p}} \prod_{j=1}^{N^-} \prod_{i=1}^{N_j^-} \Pr(I_{ij}^- = -|\mathbf{x}_{ij}^-) \tag{2-31}$$

The negative logarithm of Equation (2-31) shown in Equation (2-32) is optimized. An additional hyperparameter term, ρ, is added to control how much the negative bags affect the objective function.

$$-\ln J = -\sum_{j=1}^{N^+} \frac{1}{p} \ln\Big(\frac{1}{N_j^+} \sum_{i=1}^{N_j^+} \Pr(I_{ij}^+ = +|\mathbf{B}_j^+)^p \Big) - \rho \sum_{i=1}^{N^-} \sum_{i=1}^{N_j^-} \ln \Pr(I_{ij}^- = -|\mathbf{x}_{ij}^-) \tag{2-32}$$

In order to compute the probabilities of Equation (2-32), the instances are assumed to come from a sparse linear combination of representatives, like what is done in the FUMI algorithm (Jiao and Zare, 2015). Let $\mathbf{D}^+ = [\mathbf{d}_1^+, \ldots, \mathbf{d}_T^+]$ be the set of T target signatures, $\mathbf{D}^- = [\mathbf{d}_1^-, \ldots, \mathbf{d}_M^-]$ be the set of M background signatures, and $\boldsymbol{\alpha}_{ij}^+$ and $\boldsymbol{\alpha}_{ij}^{+b}$ be the

sparse representation of $\mathbf{x}_{ij}^{+}$, given $\mathbf{D}$ and $\mathbf{D}^{-}$, respectively. Instances that are true target instances, i.e. if $L_j = 1,\ \exists \mathbf{x}_i \in \mathbf{B}_j^{+}$, can be represented as a linear combination of the target representatives, $\mathbf{d}_t^{+}$, the background representatives, $\mathbf{d}_k^{-}$, and an error term, e_j. This is shown in Equation (2-33).

$$\mathbf{x}_i = \sum_{t=1}^{T} \alpha_{it}\mathbf{d}_t^{+} + \sum_{k=1}^{M} \alpha_{ik}\mathbf{d}_k^{-} + \varepsilon_i \text{ , s.t. } \sum_{t=1}^{T} |\alpha_{it}| \neq 0 \tag{2-33}$$

To represent negative instances, i.e. if $L_j = 0,\ \forall \mathbf{x}_i \in \mathbf{B}_j^{-}$ an instance can be represented as a linear combination of negative representatives, $\mathbf{d}_k^{-}$ and an error term, e_i. This is shown in Equation (2-34).

$$\mathbf{x}_i = \sum_{k=1}^{M} \alpha_{ik}\mathbf{d}_k^{-} + \varepsilon_i \tag{2-34}$$

To estimate the probability of an instance $\mathbf{x}_{ij}^{+}$ in $\mathbf{B}_j^{+}$ being a target point, the hybrid subpixel detector was introduced. This estimator is a ratio of distances between the instance and the target dictionary, $\mathbf{D}$, and the distances between the instance and the background dictionary, $\mathbf{D}^{-}$. This is shown in Equation (2-35).

$$\Pr(l_{ij}^{+} = +|\mathbf{B}_j^{+}) = \exp\left(-\beta \frac{||\mathbf{x}_{ij}^{+} - \mathbf{D}\boldsymbol{\alpha}_{ij}^{+}||^2}{||\mathbf{x}_{ij}^{+} - \mathbf{D}^{-}\boldsymbol{\alpha}_{ij}^{+b}||^2} \right) \tag{2-35}$$

Additionally, the probability of an instance being a negative instance is modeled as

$$\Pr(l_{ij}^{-} = -|\mathbf{x}_{ij}^{-}) = \exp(-||\mathbf{x}_{ij}^{-} - \mathbf{D}^{-}\boldsymbol{\alpha}_{ij}^{-}||^2), \tag{2-36}$$

where the probability is a function of the reconstruction error of the instance, $\mathbf{x}_{ij}^{-}$ being reconstructed as a linear combination of solely the negative dictionary elements in $\mathbf{D}^{-}$.

Lastly, solving for the sparsity vector, $\boldsymbol{\alpha}$, given a dictionary, $\mathbf{D}$, is modeled as the Lasso problem, Tibshirani (1996) and Chen et al. (2001), shown in Equation (2-37).

$$\hat{\boldsymbol{\alpha}} = \arg\min \frac{1}{2}||\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}||_2^2 + \boldsymbol{\lambda}||\boldsymbol{\alpha}||_1 \tag{2-37}$$

2.2 General Clustering Methods

Clustering techniques have been shown to be useful in many applications and often aid in extracting additional information about data that can be used in other methods. In Sections 2.2.1 - 2.2.3 a brief review of K-Means, Fuzzy C-Means (FCM), and Gaussian Mixture Models (GMM) are provided. These three clustering methods are used in the proposed multiple target multiple instance adaptive coherence estimator.

2.2.1 K-Means

The K-Means clustering algorithm clusters data in to C clusters, specified by the user. To perform clustering, the K-Means algorithm minimizes the following objective function,

$$J = \sum_{j=1}^{C} \sum_{i=1}^{N} ||\mathbf{c}_j - \mathbf{x}_i^{(j)}||^2 \tag{2-38}$$

where N is the number of samples that belong to cluster j, $\mathbf{x}_i^{(j)}$ is a sample from cluster j, and $\mathbf{c}_j$ is the center of cluster j (MacQueen et al., 1967). This objective function is assuming a Euclidean distance measure is used, but alternative distance measures can be used. By deriving the update equations for both the labels and the cluster centers it can be shown that the K-Means algorithm is a special case of the Expectation Maximization algorithm. In the E-Step, the assignment for a data point, $\mathbf{x}_i$, is determined by finding the cluster center that is closest to that data point. In the M-Step, each cluster center is updated using the data points that belong to that cluster. To update a cluster center, $\mathbf{c}_j$, the average of all of the samples that belong to that cluster is computed and set to be the new cluster center. This is shown in Equation (2-39), where N is the number of data points in the j^{th} cluster.

$$\mathbf{c}_j = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i^{(j)} \tag{2-39}$$

2.2.2 Fuzzy C-Means

The Fuzzy C-Means (FCM) clustering algorithm is similar to K-Means clustering except it uses a soft "fuzzy" membership instead of a hard, crisp membership for data points in clusters.

The algorithm minimizes the following objective function,

$$\arg\min_{C} \sum_{j=1}^{C} \sum_{i=1}^{N} \mu_{ij}^{m} ||\mathbf{c}_j - \mathbf{x}_i||^2, \quad 1 \leq m < \infty \quad \text{s.t.} \quad \sum_{j=1}^{C} \mu_{ij} = 1 \quad \forall i \tag{2-40}$$

where m is a hyperparameter known as the fuzzifier that controls the fuzziness or how soft the membership labels are (Peizhuang, 1983). The larger the fuzzifier, the more fuzzy. The smaller the fuzzifier, the more crisp FCM becomes, and the more it is like K-Means. C is the number of clusters, N is the number of instances, $\mathbf{c}_j$ is the center for the j^{th} cluster, and μ_{ij} is the membership for the i^{th} instance, $\mathbf{x}_i$, corresponding to the j^{th} cluster.

To update the cluster centers, similarly to K-Means, the weighted average of the points in a cluster are computed, where the weight is the membership of the datapoint to the j^{th} cluster. This is shown below in Equation (2-41).

$$\mathbf{c}_j = \frac{\sum_{i=1}^{N} \mu_{ij}^{m} \mathbf{x}_i}{\sum_{i=1}^{N} \mu_{ij}^{m}} \tag{2-41}$$

In Equation (2-42) the update equation for the memberships of individual data points is shown. To update the membership of the i^{th} datapoint to the j^{th} cluster we can see that it is related to the distances to the j^{th} cluster and the other clusters. Namely, the ratio of the distance to the j^{th} cluster over the sum of the distances to all other clusters raised to the $\frac{2}{m-1}$ is used to compute the membership. So this can be interpreted that the membership for the j^{th} cluster is proportional to the distance to the j^{th} cluster center c_j, over the distances to all of the other cluster centers.

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{||\mathbf{x}_i - \mathbf{c}_j||}{||\mathbf{x}_i - \mathbf{c}_k||} \right)^{\frac{2}{m-1}}} \tag{2-42}$$

2.2.3 Gaussian Mixture Model

The Gaussian Mixture Model (GMM) is a generative model that fits Gaussian distributions to data and can be used for clustering (Murphy, 2013). Like K-Means and Fuzzy C-Means,

the GMM algorithm makes use of the Expectation Maximization (EM) optimization strategy to update the parameters of the model. One of the primary differences between these is that GMM is a generative model. So instead of using distances to determine the clusters, it uses multivariate Gaussian distributions and fits them to the data using probability to form clusters. With this comes some other differences like the fact that clusters do not need to be spherical in the feature space. The distributions can have different variances along different dimensions or a full covariance. This is one of the biggest benefits to GMM and allows for data to have different variances in different dimensions and still be clustered correctly.

A Gaussian mixture model is represented as a sum of weighted Gaussian distributions over data and is shown in Equation (2-43).

$$\mathrm{P}(x) = \sum_{k=1}^{C} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_\mathrm{k}, \boldsymbol{\Sigma}_\mathrm{k}) \quad \text{s.t.} \quad \pi_k \geq 0 \quad \text{and} \quad \sum_{k=1}^{C} \pi_k = 1 \tag{2-43}$$

where π_k is the weight for the k^{th} Gaussian, and $\boldsymbol{\mu}_\mathrm{k}$ and $\boldsymbol{\Sigma}_\mathrm{k}$ are the mean and covariance for the k^{th} Gaussian distribution respectively. To simplify notation, $\boldsymbol{\theta}_\mathrm{k}$ will be used to represent the parameters $\boldsymbol{\mu}_\mathrm{k}$ and $\boldsymbol{\Sigma}_\mathrm{k}$ for the k^{th} Gaussian distribution

In order to fit the data, most often an EM approach is done to determine the update equations. EM is an approach that helps solve for complex likelihood equations. In EM, an observed data likelihood is the likelihood that would like to be solved shown in Equation (2-44).

$$L_{observed} = \sum_{i=1}^{N} \ln \sum_{k=1}^{C} \pi_k \mathrm{P}(\mathbf{x}_\mathrm{i}|\boldsymbol{\theta}_\mathrm{z}) \tag{2-44}$$

In order to solve Equation (2-44), latent variables are used and create the complete likelihood shown in Equation (2-45). Where z_i is the missing latent variable corresponding to which component the data point $\mathbf{x}_\mathrm{i}$ belongs to. Then, taking the log of the complete likelihood, simplifies Equation (2-45) to a form that can be optimized, shown in Equation (2-46).

$$L_{complete} = \ln \sum_{i=1}^{N} \mathrm{P}(\mathbf{x}_i|\boldsymbol{\theta}_z)\mathrm{P}(z_i) \tag{2-45}$$

$$= \prod_{i=1}^{N} \mathcal{N}(\mathbf{x}_i|\boldsymbol{\theta}_{zi})\pi_{zi} \tag{2-46}$$

To perform EM, two steps are done the expectation step and the maximization step. In the expectation step, the probability of a datapoint corresponding to each Gaussian component is computed. In other words, the Gaussian component a data point is expected to correspond to is computed given the current parameters of the Gaussian components. The computation for this is shown in Equation (2-47).

$$\mathrm{P}(z_i|\mathbf{x}_i, \boldsymbol{\theta}^{(t)}) = \frac{\pi_{zi}^{(t)}\mathrm{P}(\mathbf{x}_i|\boldsymbol{\theta}_{zi}^{(t)}, z_i)}{\sum_{k=1}^{C} \pi_k^{(t)}\mathrm{P}(\mathbf{x}_i|\boldsymbol{\theta}_k^{(t)}, k)} \tag{2-47}$$

Equation (2-47) shows that the updated probability of belonging to a specific component is the ratio of the weighted probability of belonging to that component over all of the other components weighted probabilities, where the weight is the current membership of the sample to that component at iteration t. The individual probabilities of belonging to a Gaussian component can be computed directly using the sample, $\mathbf{x}_i$, and the parameters of the components, $\boldsymbol{\theta}^{(t)}$, at iteration t.

In the maximization step, the Gaussian components are maximized given the current iteration's expected probability that the data points belong to a certain component. To update the Gaussian components, the means and covariances are updated for each component.
To update the means, Equation (2-48) is used. It is seen here that, very similarly to Fuzzy C-Means, the means of the distribution are computed as a ratio of the weighted mean of the data points, where the weight is the probability of a data point belonging to that component. So like FCM, it is a weighted mean, except in GMM, the weight corresponds to the probability instead of the membership associated with FCM.

$$\boldsymbol{\mu}_{\mathrm{k}} = \frac{\sum_{i=1}^{N} \mathrm{P}(z_i = k|\mathbf{x}_{\mathrm{i}}, \boldsymbol{\theta}^{(t)})\mathbf{x}_{\mathrm{i}}}{\sum_{i=1}^{N} \mathrm{P}(z_i = k|\mathbf{x}_{\mathrm{i}}, \boldsymbol{\theta}^{(t)})} \tag{2-48}$$

To update the covariances of the Gaussian components, the derivation of the update equation shows a very similar result as the means. The covariance update equation, shown in Equation (2-49), shows that the updated covariance for the k^{th} component is the covariance matrix of the data with respect to the mean of the component, weighted by the ratio of the probability of those data points belonging to the k^{th} component over the probability of belonging to all of the components.

$$\boldsymbol{\Sigma}_{\mathrm{k}} = \frac{\sum_{i=1}^{N} \mathrm{P}(z_i = k|\mathbf{x}_{\mathrm{i}}, \boldsymbol{\theta}^{(\mathrm{t})})(\mathbf{x}_{\mathrm{i}} - \boldsymbol{\mu}_{\mathrm{k}})(\mathbf{x}_{\mathrm{i}} - \boldsymbol{\mu}_{\mathrm{k}})^{T}}{\sum_{i=1}^{N} \mathrm{P}(z_i = k|\mathbf{x}_{\mathrm{i}}, \boldsymbol{\theta}^{(\mathrm{t})})} \tag{2-49}$$

Lastly, during the maximization step, the latent variables must also be updated. By using the constraints $\pi_k \geq 0$ and $\sum_{k=1}^{C} \pi_k = 1$ a Lagrangian problem can be solved for the latent variables. If the Lagrangian is solved and the corresponding Lagrangian multiplier is substituted in, the resulting solution shown in Equation (2-50), is obtained.

$$\pi_k = \frac{\sum_{i=1}^{N} \mathrm{P}(z_i = k|\mathbf{x}_{\mathrm{i}}, \boldsymbol{\theta}^{(\mathrm{t})})}{\sum_{k=1}^{C}\sum_{i=1}^{N} \mathrm{P}(z_i = k|\mathbf{x}_{\mathrm{i}}, \boldsymbol{\theta}^{(\mathrm{t})})} \tag{2-50}$$

This shows that the weight of each Gaussian component is updated by computing the probability of all of the data points belonging to that component divided by the probability of all of the data points belonging to all of the components.

2.2.4 Summary

With the literature of these three methods provided, it can be seen that these three clustering approaches cluster data very similarly. It is seen that Fuzzy C-Means is an extension of K-Means where the cluster memberships are no longer crisp, but rather soft,

fuzzy memberships. Both K-Means and Fuzzy C-Means are solved with an iterative approach that performs alternating optimization very similar to the EM optimization strategy that the Gaussian Mixture Model makes use of. Finally, it can be seen that the Gaussian Mixture Model is expected to perform similarly to Fuzzy C-Means, except that it can allow for clusters to be ellipsoidal instead of spherical due to the addition of the covariance in the GMM algorithm. This shows that although Fuzzy C-Means is not a probabilistic model while GMM is, they are computed very similarly except that FCM requires spherical clusters.

2.3 MIL Clustering for Dictionary Learning

In some applications of target detection, a target signature may not have one form across different conditions. For example, this can occur in hyperspectral data due to variations in lighting conditions or inner class variations. This can also occur in the Army subsurface explosive hazard dataset that I worked on throughout my research at the University of Florida. Explosive hazards can vary in size, orientation, depths, and surrounding soil type. This induces a need to learn multiple signatures. The proposed methods in this document will attempt to tackle this problem.

A number of the proposed techniques utilize clustering before initialization to obtain additional information about the data. By clustering the data, the proposed method, multi target MI-ACE/SMF, will have additional information about the data and how the bags relate to each other. In Sections 2.3.1 - 2.3.3 a number of multiple instance learning and multiple instance regression clustering techniques are reviewed, some of which are used in the proposed methods.

2.3.1 Multiple Instance Cluster Regression

Multiple Instance Cluster Regression (MI-ClusterRegress) uses clustering to aid in developing regression models for a multiple instance regression (MIR) problem (Wagstaff et al., 2008). In MIR, the data format is similar to MIL, except that instead of a bag of data corresponding to either a positive or negative label, a particular bag corresponds to a particular

value. So the goal of MIR is to train a regression model, or multiple regression models, that can predict the labels, particular values, of new bags from their contents.

MI-ClusterRegress attempts to create a structured form of the data so that regression can be applied using individual samples with corresponding scalars. MI-ClusterRegress assumes that the data in each bag comes from a number of distinct distributions. With this assumption, MI-ClusterRegress is able to learn the distributions through clustering, and then apply a regression model to each distribution. By clustering the data, this approach is able to map the instances in the bags to a particular distribution. Then these memberships can be used to create exemplar points which in return can be used to develop individual regression model for each distribution. Without the addition of the clustering, it can be difficult to determine the correct scalar label for each instance in a bag because the instance level label is not provided in the MIR framework.

The algorithm's main assumption and what helps solve this problem is that the instances in each bag are drawn from underlying distributions that can be clustered. It is also assumed that one cluster is responsible for each bag's regression label. To solve this problem, the instances from each bag are reduced to exemplar points which are the representation points of that bag's instances for a particular cluster. So if there are C clusters, each bag will have C exemplar points, one for each cluster. These exemplar points are then used for training regression models.

An exemplar point, $\hat{\mathbf{B}}_j^k$, is a weighted average of the instances in a bag, where the weights correspond to the membership of that instance to the corresponding cluster. Namely, the exemplar point for cluster k within bag j, denoted as $\hat{\mathbf{B}}_j^k$, is the average of all items in bag j weighted by their memberships in cluster k, denoted by R. Then, using the exemplars, a regression model is fit to each cluster k, using the exemplar points from each bag that correspond to cluster k. To determine the membership relevance, R, for each instance i, Equation (2-51), (2-52), and (2-53) are used.

$$r_i = \mathrm{P}(\mathbf{x}_\mathrm{i} \in c_k | \mathbf{B}_\mathrm{j}; \boldsymbol{\theta}_{\mathrm{c_k}}), \forall i \tag{2-51}$$

$$z := \sum_{i=1}^{N_j^+} r_i \tag{2-52}$$

$$R_i := \frac{r_i}{z}, \forall i \tag{2-53}$$

Here, $\mathbf{x}_\mathrm{i} \in c_k$ means that instance $\mathbf{x}_\mathrm{i}$ was generated by the distribution associated with cluster c_k. This probability in Equation (2-51) can be computed using the learned parameters, $\boldsymbol{\theta}_\mathrm{k}$, from each of the Gaussian mixture model distributions. Then a normalization term z is computed for the bag, $\mathbf{B}_\mathrm{j}$, as the sum of all memberships from bag, $\mathbf{B}_\mathrm{j}$. Here, N_j^+ is the number of instances in bag j. Lastly, each instance's membership is normalized by z to form the relevance, R_i, of an instance belonging to the k^{th} distribution in bag j. In the original paper, the authors chose to use a Gaussian mixture model for clustering, but the only requirement for clustering is that the clusters produce C generative models that can produce a membership for all of the instances.

The overall algorithm psuedocode has been provided below:

Algorithm 2 MI-ClusterRegress Algorithm

Inputs: Bagged Data $\mathbf{B}_{\mathrm{j}=1,\ldots,\mathrm{N}}$, labels $\mathbf{Y}$, number of clusters C
Outputs: Regression parameters $\boldsymbol{\gamma}'$ and cluster parameters $\boldsymbol{\theta}'$

$\mathbf{X} := \cup_{i=1,\ldots,N}(\mathbf{B}_\mathrm{j})$ //Group all data together regardless of bag structure
$\boldsymbol{\theta}_{\mathrm{i}=1,\ldots,\mathrm{C}} :=$ Cluster($\mathbf{X}$, C) //Cluster all items into C clusters
for $j = 1$ to N **do**
    for $k = 1$ to C **do**
        $R :=$ Relavance($\mathbf{B}_\mathrm{j}, \boldsymbol{\theta}_\mathrm{k}$) //Compute memberships for instances in $\mathbf{B}_\mathrm{j}$
        $\hat{\mathbf{B}}_\mathrm{j}^\mathrm{k} := \mathbf{B}^\mathrm{j} R$ //Construct Exemplar for bag $\mathbf{B}^\mathrm{j}$ in cluster k
    end for
end for
for $k = 1$ to C **do**
    $\boldsymbol{\gamma}_\mathrm{k} :=$ Regress($\{\hat{\mathbf{B}}_\mathrm{j}^\mathrm{k}\}_{J=1,\ldots,N}, \mathbf{Y}$) //Regression model for cluster k
end for
$[\boldsymbol{\gamma}', \boldsymbol{\theta}'] :=$ Select($\{\boldsymbol{\gamma}_\mathrm{k}, \boldsymbol{\theta}_\mathrm{k}\}_{k=1,\ldots,C}, \mathbf{Y}$) //Select best local model that maps bags to labels

2.3.2 Fuzzy Clustering of Multiple Instance Data

Fuzzy Clustering of Multiple Instance Data (FCMI) makes use of the previous work done by Maron and Lozano-Pérez (1998), and extends it to identify dense regions of the feature space with maximal correlation to positive instances and minimal correlation to negative instances (Karem and Frigui, 2015), (Karem and Frigui, 2016). By creating a new Multi-target concept Diverse Density metric (MDD), FCMI is able to identify multiple target concepts simultaneously. Namely, the objective of FCMI is to identify K target concepts, $\mathbf{T} = \{\mathbf{t}_1, ..., \mathbf{t}_k, ...\mathbf{t}_K\}$ that correspond to regions of the features space with as many positive instances and as few negative instances as possible. The method assumes there are N bags, $\mathbf{B} = \{\mathbf{B}_1, ..., \mathbf{B}_j, ...\mathbf{B}_N\}$, where each bag is assumed to have a label as positive or negative. Where a positive labeled bag, $\mathbf{B}^+ = \{\mathbf{B}_1^+, ..., \mathbf{B}_j^+, ..., \mathbf{B}_N^+\}$, corresponds to a bag having at least one instance from the true target class, and a negative labeled bag, $\mathbf{B}^- = \{\mathbf{B}_1^-, ..., \mathbf{B}_j^-, ..., \mathbf{B}_N^-\}$, corresponds to a bag with no instances in it that correspond to the true target class. In each of the bags, signified by j, there are an unknown number of instances; N_j^+ designates the number of positive instances in bag j, and N_j^- designates the number of negative instances in bag j. Each instance, $\mathbf{x}_i$ is signified by i. $\mathbf{X}_j = \{\mathbf{x}_{j1}, ..., \mathbf{x}_{ji}, ..., \mathbf{x}_{jN^+}\}$, and each instance is a D-dimensional feature vector, $\mathbf{x}_{ji} = \{\mathbf{x}_{ji1}, ..., \mathbf{x}_{jid}, ..., \mathbf{x}_{jiD}\}$. To accomplish this, it is assumed that each bag, $\mathbf{B}_j$, belongs to each target concept, $\mathbf{t}_k$, with a fuzzy membership μ_{jk} provided the constraints in Equation (2-54).

$$\mu_{jk} \in [0, 1] \quad and \quad \sum_{k=1}^{K} \mu_{jk} = 1 \tag{2-54}$$

Let $\mathbf{U} = [\mu_{jk}]$ be the set of all memberships for all target bag pairs, namely for $k = 1, ..., T$ and $j = 1, ..., N$. The Multi-target concept Diverse Density (MDD) metric is defined as

$$\text{MDD}(\mathbf{T}, \mathbf{U}) = \prod_{j=1}^{N} \prod_{K=1}^{K} \mu_{jk}^m \text{Pr}(\mathbf{t}_k | \mathbf{B}_j) \tag{2-55}$$

where m is a parameter known as the fuzzifier that controls the fuzziness of the clusters like in FCM (Peizhuang, 1983). The FCMI algorithm seeks to find the optimal $(\mathbf{T}, \mathbf{U})$ that maximizes the MDD metric in Equation (2-55).

Instead of maximizing Equation (2-55), the negative log-likelihood is taken and minimized.

$$J(\mathbf{T}, \mathbf{U}) = -\ln(\text{MDD}(\mathbf{T}, \mathbf{U})) = \sum_{j=1}^{N} \sum_{k=1}^{K} \mu_{jk}^{m} \{-\ln(\Pr(\mathbf{t}_k|\mathbf{B}_j))\} \tag{2-56}$$

To minimize Equation (2-56) with respect to $\mathbf{U}$, Lagrangian multipliers, $\mathbf{\Lambda}$, are used with the second constraint from Equation (2-54). This is shown in Equation (2-57).

$$J(\mathbf{T}, \mathbf{U}, \mathbf{\Lambda}) = \sum_{j=1}^{N} \sum_{k=1}^{K} \mu_{jk}^{m} \{-\ln(\Pr(\mathbf{t}_k|\mathbf{B}_j))\} - \lambda_j \left(\sum_{k=1}^{K} \mu_{jk} - 1 \right) \tag{2-57}$$

Then, assuming the partial densities $\Pr(\mathbf{t}_k|\mathbf{B}_j)$ and the columns of $\mathbf{U}$ are independent for all bags $j = 1, ..., N$, Equation (2-57) can be reduced to N minimization problems shown in Equation (2-58), where λ_j is the lagrangian multiplier for the j^{th} lagrangian problem.

$$J_j(\mathbf{T}, \mathbf{U}_j, \lambda_j) = \sum_{k=1}^{K} \mu_{jk}^{m} \{-\ln(\Pr(\mathbf{t}_k|\mathbf{B}_j))\} - \lambda_j \left(\sum_{k=1}^{K} \mu_{jk} - 1 \right), \quad j = 1, ..., N \tag{2-58}$$

Finally, by taking the partial derivative of Equation (2-58) with respect to μ_{jk}, setting it to 0, and solving for μ_{jk}, the update equation for μ_{jk} can be obtained. This leads to the following update equation shown in Equation (2-59) for μ_{jk}.

$$\mu_{jk} = \frac{-\ln(\Pr(\mathbf{t}_k|\mathbf{B}_j))^{\frac{1}{m-1}}}{\sum_{k=1}^{K} -\ln(\Pr(\mathbf{t}_k|\mathbf{B}_j))^{\frac{1}{m-1}}} \tag{2-59}$$

To determine the optimal target concepts, $\mathbf{T}$, FCMI uses the multiple instance framework and the often used NOISY-OR model (Srinivas, 1993). This can be modeled as such:

$$\Pr(\mathbf{t}_k|\mathbf{B}_j) = \begin{cases} 1 - \prod_{i=1}^{N_j^+}(1 - \Pr(\mathbf{x}_{ji} \in \mathbf{t}_k)) & \text{if} \quad \text{Label}(\mathbf{B}_j) = 1 \\ \prod_{i=1}^{N_j^-}(1 - \Pr(\mathbf{x}_{ji} \in \mathbf{t}_k)) & \text{if} \quad \text{Label}(\mathbf{B}_j) = 0 \end{cases} \tag{2-60}$$

The $\Pr(\mathbf{x}_{ji} \in \mathbf{t}_k)$ can be regarded as the similarity of instance $\mathbf{x}_{ji}$ to $\mathbf{t}_k$. Assuming the centroids, $\mathbf{c}_k$, of the fuzzy cluster represent the target concepts, $\mathbf{t}_k$, the probability can be expressed as

$$\Pr(\mathbf{x}_{ji} \in \mathbf{t}_k) = \exp\left(\sum_{d=1}^{D} \mathbf{s}_{kj}(\mathbf{x}_{jid} - \mathbf{c}_{kj})^2\right) \tag{2-61}$$

where $\mathbf{s}_{kj}$ is a scaling vector that weights the individual features of $\mathbf{t}_k$ and $\mathbf{x}_{jk}$ so that the features that are responsible for discriminating between target and non-target instances are weighted more heavily and the features that are irrelevant for discriminating have a low weight. To find the optimal target concepts the optimal cluster centers, $\mathbf{c}_k$, and their respective scaling vector, $\mathbf{s}_{kj}$, must be determined. This can be accomplished by holding the memberships fixed, taking the derivative with respect to the cluster centers and with respect to the scaling vectors, setting the derivative to zero, and solving for the cluster centers and scaling vectors respectively. When this is done, the derivation for the update equations for $\mathbf{c}_k$ and $\mathbf{s}_{kj}$ are shown in Equation (2-62) and (2-63).

$$\frac{\partial J}{\partial \mathbf{c}_k} = -\sum_{j=1}^{N} \frac{\mu_{jk}^m}{\Pr(\mathbf{t}_k|\mathbf{B}_k)} \times \frac{\partial \Pr(\mathbf{t}_k|\mathbf{B}_j)}{\partial \mathbf{c}_k} = 0 \tag{2-62}$$

$$\frac{\partial J}{\partial \mathbf{s}_k} = -\sum_{j=1}^{N} \frac{\mu_{jk}^m}{\Pr(\mathbf{t}_k|\mathbf{B}_j)} \times \frac{\partial \Pr(\mathbf{t}_k|\mathbf{B}_j)}{\partial \mathbf{s}_k} = 0 \tag{2-63}$$

To evaluate the probabilities in Equation (2-62) and (2-63), the noisy or model shown in Equation (2-60) must be considered. The $\Pr(\mathbf{t}_k|\mathbf{B}_j)$ depends on whether a bag is positive or negative, and therefore the optimal clusters and scales will also depend on whether the corresponding bag is positive or negative. This is expanded out and shown in Equation (2-64) and (2-65).

$$\frac{\partial J}{\partial \mathbf{c}_k} = -\sum_{j=1}^{N+} \frac{\mu_{jk}^m}{\Pr(\mathbf{t}_k|\mathbf{B}_j^+)} \times \frac{\partial \Pr(\mathbf{t}_k|\mathbf{B}_j^+)}{\partial \mathbf{c}_k} - \sum_{j=1}^{N-} \frac{\mu_{jk}^m}{\Pr(\mathbf{t}_k|\mathbf{B}_j^-)} \times \frac{\partial \Pr(\mathbf{t}_k|\mathbf{B}_j^-)}{\partial \mathbf{c}_k} \tag{2-64}$$

$$\frac{\partial J}{\partial \mathbf{s}_k} = -\sum_{j=1}^{N+} \frac{\mu_{jk}^m}{\Pr(\mathbf{t}_k|\mathbf{B}_j^+)} \times \frac{\partial \Pr(\mathbf{t}_k|\mathbf{B}_j^+)}{\partial \mathbf{s}_k} - \sum_{j=1}^{N-} \frac{\mu_{jk}^m}{\Pr(\mathbf{t}_k|\mathbf{B}_j^-)} \times \frac{\partial \Pr(\mathbf{t}_k|\mathbf{B}_j^-)}{\partial \mathbf{s}_k} \tag{2-65}$$

Equation (2-64) and (2-65) do not have closed-form solutions. Instead of solving for these updates directly, an approximated solution using an iterative line search approach is used to determine the parameter updates. The algorithm pseudocode is provided in Algorithm 3. $\mathbf{B}^+$ and $\mathbf{B}^-$ are the sets of positive and negative bags respectively, K is the number of desired targets, $\mathbf{C}$ are the centers of K target concepts, $\mathbf{S}$ are the scales of the K target concepts, and $\mathbf{U}$ are the memberships of all bags in all target concepts.

Algorithm 3 FCMI Algorithm

Inputs: $\mathbf{B}^+$, $\mathbf{B}^-$, K
Outputs: $\mathbf{C}$, $\mathbf{S}$, $\mathbf{U}$

1: Initialize $\mathbf{c}^k$ and $\mathbf{s}^k$ for $k = 1, ..., K$
2: **repeat**
3: Update μ_{jk} using Equation (2-59)
4: Update $\mathbf{C}$ and $\mathbf{S}$ using a line search algorithm that minimizes eq. (2-62) and (2-63)
5: **until** centers do not change or max number of iterations reached
6: **return** $\mathbf{C}$, $\mathbf{S}$, $\mathbf{U}$

2.3.3 Robust Fuzzy Clustering for Multiple Instance Linear Regression

The Robust Fuzzy Clustering for Multiple Instance Linear Regression (RFC-MILR) algorithm uses various clustering techniques to add in multiple instance regression problems. First, fuzzy clustering is used to determine the primary instances of a bag, and secondly possibilistic clustering is used to identify non-primary instances of a bag (Trabelsi and Frigui, 2018). The combined features and labels are both used for clustering and help to identify multiple local linear regression models. This algorithm has three main steps. First, shown in Section 2.3.3.1, fuzzy clustering is used to obtain a membership degree for the instances in

each bag to fit local linear regression models. Second, shown in Section 2.3.3.2, a possibilistic robust clustering method is used to minimize the affect that non-primary instances have on the local linear regression models. Non-primary instances are those that are not responsible for the label of the bag. Lastly, shown in Section 2.3.3.3, the properties of the possibilistic model are used to determine the optimal number of regression models.

2.3.3.1 Initial regression models using fuzzy clustering

The first step of the RFC-MILR is to use Fuzzy C-Means (FCM) to determine membership values for each sample in a positive bag to obtain an initial estimate of the local linear regression models. FCM is used to obtain the resulting memberships to identify the primary instances to train an initial regression model. The generic objective function for FCM with any distance metric is written as

$$J = \sum_{j=1}^{C} \sum_{i=1}^{N} (\mu_{ij})^m dist_{ij}^2, \tag{2-66}$$

which is subject to the constraints in Equation (2-67), and has been previously reviewed in Section 2.2.2.

$$\mu_{ij} \in [0,1] \ \forall i \text{ and } \forall j; \quad \text{and} \quad \sum_{j=1}^{C} \mu_{ij} = 1 \ \forall i \tag{2-67}$$

In the originally proposed RFC-MILR, the distance metric, $dist_{ij}^2$, between the cluster centers and a sample is shown in Equation (2-68), where $\mathbf{x}'_{\text{ji}} = [\mathbf{x}_{\text{ji}}, \mathbf{y}_{\text{j}}] \in \mathbb{R}^{D+1}$. Any distance could be used, but Trabelsi and Frigui (2018) decided to use the distance in Equation (2-68) to give more importance to distances projected on to eigenvectors associated with smaller eigenvalues.

$$dist_{ij}^2 = \sum_{d=1}^{D+1} v_{jk} ((\mathbf{c}_{\text{j}} - \mathbf{x}'_{\text{ji}}) \cdot \mathbf{e}_{\text{jd}})^2 \tag{2-68}$$

In Equation (2-68), $\mathbf{c}_{\text{j}}$ is the center of cluster j, $\mathbf{e}_{\text{jd}}$ is the d^{th} unit eigenvector of the covariance matrix, $\mathbf{\Sigma}_{\text{j}}$, of cluster j. Here, v_{jk} corresponds to a weighting vector for each dimension that controls the amount of distance that the d^{th} dimension contributes to the total distance. It is computed as the product of all eigenvalues divided by the eigenvalue

corresponding to the d^{th} dimension, shown in Equation (2-69). The distance is proportional to the eigenvalue of each eigenvector that the data is projected on to.

$$v_{jk} = \frac{\left[\prod_{d=1}^{D+1} \lambda_{jd}\right]^{\frac{1}{D+1}}}{\lambda_{jk}} \tag{2-69}$$

To optimize FCM, an iterative algorithm is performed to alternately update the memberships, μ_{ij}, and the cluster parameters, $\mathbf{c}_j$ and $\mathbf{\Sigma}_j$. To update the cluster parameters, Equation (2-70) and (2-71) are used.

$$\mathbf{c}_j = \frac{\sum_{i=1}^{N} (\mu_{ij})^m \mathbf{x}_i}{\sum_{i=1}^{N} (\mu_{ij})^m} \tag{2-70}$$

$$\mathbf{\Sigma}_j = \frac{\sum_{i=1}^{N} (\mu_{ij})^m (\mathbf{c}_j - \mathbf{x}_i)(\mathbf{c}_j - \mathbf{x}_i)^T}{\sum_{i=1}^{N} (\mu_{ij})^m} \tag{2-71}$$

These are the same update equations as shown in Section 2.2.2 for FCM using Euclidean distance, but now a covariance, $\mathbf{\Sigma}_j$, is used because of the distance metric selected in FRC-MILR. The analysis of Equation (2-70) and (2-71) are the same as in Section 2.2.2, except a different distance metric is used. So it can be seen that the cluster center, $\mathbf{c}_j$, is still the weighted mean of the data, $\mathbf{x}_i$, weighted by their corresponding membership, $(\mu_{ij})^m$, to cluster j. As well, the covariance for cluster j is calculated as the weighted average of the covariance matrix corresponding to cluster j. FCM is computed on all of the data, for a few number of iterations, to initialize the clusters and obtain initial regression models. The regression models will include all of the data, even instances that are non-primary instances.

2.3.3.2 Non-primary instances using possibilistic clustering

The second main step of RFC-MILR is to use Possibilistic C-Means (PCM) to determine the non-primary instances, the outliers of the bag that do not correspond to the bag's label to refine the initialized models. PCM allows the membership of the samples that are considered

to be outliers to be close to 0 and the samples that correspond to inliers be close to 1, because the membership values in PCM are not constrained to sum to 1 across all clusters, $\mu_{ij} \in [0, 1]$. Through this property the desired regression models can be obtained by weighting the training data of the regression models by their respective possibilistic membership value. It can be seen that PCM has this property by relaxing the constraints of FCM shown in Equation (2-67), by having the following objective function which is minimized,

$$J = \sum_{j=1}^{C} \sum_{i=1}^{N} (\mu_{ij})^m dist_{ij}^2 + \sum_{j=1}^{C} \eta_j \sum_{i=1}^{N} (1 - \mu_{ij})^m. \tag{2-72}$$

In Equation (2-72), the η_j parameter is set as a prior to control the cluster resolution, or alternatively could be updated in each iteration using the distribution of the data in each cluster.

Like FCM, PCM uses the iterative optimization approach to update the 3 parameters of the model, μ_{ij}, $\mathbf{c}_j$ and $\mathbf{\Sigma}_j$. Similarly to how the update equations are derived for FCM, the update equations can be derived for PCM. The update equation for μ_{ij} is shown below in Equation (2-73).

$$\mu_{ij} = \left[1 + \left(\frac{dist_{ij}^2}{\eta_i} \right)^{\frac{1}{m-1}} \right]^{-1} \tag{2-73}$$

The update equations for the cluster parameters, $\mathbf{c}_j$ and $\mathbf{\Sigma}_j$, are derived to be the same as for the fuzzy case, shown in Equation (2-70) and (2-71). It can be seen that Equation (2-70) and (2-71) are the same update equations used for FCM. This is because the membership constraint only affects the update equation for memberships. The membership constraints do not affect the parameters of the clusters, only the membership values are used to update the parameters of the clusters, $\mathbf{c}_j$ and $\mathbf{\Sigma}_j$.

Once the cluster centers and covariances no longer change more than some stopping threshold, PCM has converged.

2.3.3.3 Optimal number of regression models

A technique is then used to merge similar clusters by taking advantage of a useful property of PCM. Since PCM does not constrain the memberships to sum to 1, there can be several similar or identical clusters. The authors initialize an over estimate of clusters needed for regression, and then after PCM has converged, this property is exploited to merge similar clusters. To merge two clusters m and n the following equation is used, where Θ_M is a merging threshold hyperparameter.

$$\frac{\sum_{i=1}^{N} |\mu_{im} - \mu_{in}|}{\sum_{i=1}^{N} |\mu_{im}| + \sum_{i=1}^{N} |\mu_{in}|} < \theta_M \tag{2-74}$$

It is assumed that the underlying regression model is linear and only requires one cluster to appropriately perform regression. So if the algorithm produces more than one cluster after merging has taken place, an optimal cluster, p, is selected using Equation (2-75). Here C' is the number of clusters that remain after merging and ε_j is the sum of distances between the j^{th} cluster center and all of the samples weighted by their membership.

$$p = \arg \min_{j=1,\ldots,C'} \left\{ \varepsilon_j = \sum_{i=1}^{N} (\mu_{ij})^m dist_{ij}^2 \right\} \tag{2-75}$$

Then, the primary instances of cluster p are extracted using Equation (2-76) and stored in P. Here, the set of primary instances, $\mathbf{P}$, are the inliers of cluster p. These instances are determined by checking if the membership, μ_p, of instance $\mathbf{x}_\text{i}$ is greater than a hyperparameter threshold θ_p which is typically set to 0.1.

$$\mathbf{P} = \{\mathbf{x}_\text{i}, i = 1, \ldots, N \mid \mu_p > \theta_p\} \tag{2-76}$$

These instances, $\mathbf{x} \in \mathbf{P}$, as well as the parameters of the cluster, $\mathbf{c}_\text{p}$ and $\mathbf{\Sigma}_\text{p}$, are used to determine the final regression model parameters. Let $\mathbf{x} = [x_1, \ldots, x_d, y] \in \mathbf{P}$ and $\mathbf{e}_\text{min} =$

$[e^1_{min}, ..., e^{d+1}_{min}]$ be the eigenvector associated with the smallest eigenvalue, λ_{min} of $\mathbf{\Sigma}_p$. Since $\mathbf{x}$ belongs to cluster p, this leads to the property that

$$\mathbf{e}_{\text{min}} \cdot \mathbf{x} = \mathbf{e}_{\text{min}} \cdot \mathbf{c}_p \ . \tag{2-77}$$

Decomposing $\mathbf{x}$ into it's feature vector and its corresponding label y, Equation (2-78) is obtained.

$$e^{D+1}_{min} y + \sum_{d=1}^{D} e^d_{min} x_d = \mathbf{e}_{\text{min}} \cdot \mathbf{c}_p \tag{2-78}$$

Then, solving for y provides the solution to the regression model for the optimal cluster, p.

$$y = f(x) = \frac{\mathbf{e}_{\text{min}} \cdot \mathbf{c}_{\text{opt}}}{e^{D+1}_{min}} - \sum_{d=1}^{D} \frac{e^d_{min}}{e^{D+1}_{min}} x_d \tag{2-79}$$

This regression model is used for predicting labels for unknown bags, $\mathbf{B}^t = [\mathbf{x}^t_1, ..., \mathbf{x}^t_N]$. To perform prediction, first, it is determined which primary instance in $\mathbf{P}$ is closest to all of test instances, $\mathbf{x}^t_n$. The label of the primary instance that is determined to be the closest to the test instance is assumed to be a good initial estimate for the label of the test instance, $\mathbf{x}^t_n$. This initial estimate is used for all of the instances in the test bag, and PCM is run to determine the possibilistic memberships of the test instances. The test instance with the largest possibilistic membership is determined to be the primary instance of the test bag. This is modeled by Equation (2-80).

$$\mathbf{x}^t_{\text{prim}} = \{\mathbf{x}^t_k \mid \mu_k = \max_{i=1...N} \{\mu_i\}\} \tag{2-80}$$

Finally, a test bag, $\mathbf{B}^t$, is labeled using the regression value of the primary instance, $y(\mathbf{B}^t) = f(\mathbf{x}^t_{\text{prim}})$. If desired, more than one primary instance can be used to label a test bag. The same process would be done, but the label of the test bag, $\mathbf{B}^t$, would become the average of all of the selected primary instances' regression values.

CHAPTER 3
PROPOSED METHODS

3.1 Multi-Target Multiple Instance Adaptive Cosine Estimator/Spectral Match Filter

The objective of the Multi-Target Multiple Instance Adaptive Cosine Estimator (Multi-Target MI-ACE) and Multi-Target Multiple Instance Spectral Match Filter (Multi-Target MI-SMF) algorithms is to learn a dictionary of multiple target representations, focusing on maximizing detection of those targets against a background class. This algorithm fits the Multiple Instance Learning (MIL) framework proposed by Dietterich et al. (1997) and assumes the data is grouped in to bags with bag level labels. With this, let $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ be training data with each sample, $\mathbf{x}_i$ being a vector with dimensionality D. The data is grouped into J bags $\mathbf{B} = \{\mathbf{B}_1, ..., \mathbf{B}_J\}$ with labels, $L = \{L_1, ..., L_J\}$, where $L_j \in \{0, 1\}$.

A bag is considered positive, $\mathbf{B}_j^+$, with label, $L_j = 1$, when there exists at least one instance, $\mathbf{x}_{ji}$, in bag j that is from the target class, $l_{ji} = 1$, seen in Equation (3-1). Additionally, a bag is considered negative, $\mathbf{B}_j^-$, with label $L_j = 0$, if all instances in bag j are from the background class, $l_{ji} = 0$, seen in Equation (3-2). The number of instances in both positive and negative bags is variable.

$$\text{if} \quad L_j = 1, \quad \exists \mathbf{x}_{ji} \in \mathbf{B}_j^+ \quad \text{s.t.} \quad l_{ji} = 1 \tag{3-1}$$

$$\text{if} \quad L_j = 0, \quad \forall \mathbf{x}_{ji} \in \mathbf{B}_j^-, \quad l_{ji} = 0 \tag{3-2}$$

The objective function of the original MI-ACE algorithm (Zare et al., 2018) shown in Equation (3-3) has been extended to include multiple target signatures in a dictionary, $\mathbf{S}$, as shown in Equation (3-4).

$$\arg\max_{\mathbf{s}} \frac{1}{N^+} \sum_{j:L_j=1} D(\mathbf{x}_j^*, \mathbf{s}) - \frac{1}{N^-} \sum_{j:L_j=0} \frac{1}{N_j^-} \sum_{\mathbf{x}_i \in B_j^-} D(\mathbf{x}_i, \mathbf{s}) \tag{3-3}$$

$$\max_{\mathbf{S}} \quad \frac{1}{N^+} \sum_{j:L_j=1} \max_{s_k \in \mathrm{S}} (D(\mathbf{x}^*_{j,k}, \mathbf{s}_\mathrm{k})) - \frac{1}{N^-} \sum_{j:L_j=0} \frac{1}{N_j^-} \sum_{x_i \in B_j^-} \max_{s_k \in \mathrm{S}} (D(\mathbf{x}_i, \mathbf{s}_k)) \tag{3-4}$$

Here N^+ and N^- are the number of positive and negative bags respectively, N_j^- is the number of instances in negative bag j, and $\mathbf{s}_\mathrm{k}$ is the k^{th} target signature in the dictionary, $\mathbf{S}$, shown in Equation (3-5).

$$\mathbf{S} = \begin{bmatrix} \mathbf{s}_1 & \mathbf{s}_2 & \dots & \mathbf{s}_\mathrm{K} \end{bmatrix} \tag{3-5}$$

$\mathbf{x}^*_{\mathrm{j,k}}$ is known as the positive bag representative and is the instance in the j^{th} positive bag that is the most similar to the k^{th} estimated target signature, $\mathbf{s}_\mathrm{k}$. This is shown in Equation (3-6).

$$\mathbf{x}^*_{j,k} = \arg \max_{x_i \in B_j^+} D(\mathbf{x}_i, \mathbf{s}_k) \tag{3-6}$$

$D(\mathbf{x}, \mathbf{s})$ represents the similarity between two feature vectors $\mathbf{x}$ and $\mathbf{s}$. The similarity metrics used are the Adaptive Cosine Estimator (ACE) and the Spectral Match Filter (SMF), which have been reviewed in Section 2.1.5. ACE and SMF are both whitened by the background and normalized by the signatures being compared. ACE normalizes the inner product by the magnitude of both signatures, whereas SMF only normalizes the inner product by the target signature. Moving forward, the use of the double hat will signify a whitened and normalized target signature, $\hat{\hat{\mathbf{s}}}$, or sample, $\hat{\hat{\mathbf{x}}}$ using ACE. The use of a single hat will signify a whitened and normalized target signature, $\hat{\mathbf{s}}$, or sample, $\hat{\mathbf{x}}$ using SMF.

The objective function has been modified to include multiple target signatures. By using a max operation across the detection of the target signatures and positive bag representatives in the first term, this allows each target signature to focus on classifying a particular variation or target type of interest. This way, the learned target signatures can fill in the total target class data space and each target signature can focus on estimating an individual target subspace of interest instead of trying to learn a single target representative for all target variations. Additionally, the max operation in the second term was used for consistency and to penalize the algorithm for learning any target signature that is like the background.

The Multi-Target MI-ACE and Multi-Target MI-SMF algorithms have two primary steps, first K target signatures are initialized and then the initialized signatures are optimized over the training data. Three categories of initialization techniques are proposed and included in Sections 3.2 - 3.4 known as the greedy approach, the uniqueness term approach, and the clustering approach. Additionally, two optimization techniques follow, the weighted optimization approach and the uniqueness term approach.

3.2 Greedy Initialization Approach

To learn the dictionary of targets, $\mathbf{S}$, the naive brute force approach would be to try every possible instance from the positive bags, referred to as target candidates, and determine which combination of those target candidates would maximize the objective function. This approach is unfeasible though because as the number of targets, K, increases, the algorithm complexity grows rapidly. This approach would grow rapidly with the number of targets, K, at a rate of $\binom{N_I^+}{K}$, where N_I^+ is the total number of positive instances. This is because every combination of target candidates must be tested. As well the positive bag representatives must be determined for each of the target candidates to calculate the objective function. This operation alone is $O(N_I^+)$ and would need to be computed for every target candidate, which makes this process $O((N_I^+)^2)$. When you combine the computation complexity of computing the objective function with how many times this would need to occur, the complexity becomes $O\left(\binom{N_I^+}{k}\left(N_I^+\right)^2\right)$. This makes this approach unfeasible and therefore other approaches have been explored to get around this. The algorithm pseudocode for the Greedy approach is provided in Algorithm 4.

Algorithm 4 Greedy Initialization Method

Compute $\boldsymbol{\mu}_{\text{b}}$ and $\boldsymbol{\Sigma}_{\text{b}}$
Subtract the background mean and whiten all instances, $\hat{\mathbf{x}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{U}^{T}(\mathbf{x} - \boldsymbol{\mu}_{\text{b}})$
if Multi-Target MI-ACE **then**
    Normalize: $\hat{\hat{\mathbf{x}}} = \frac{\hat{\mathbf{x}}}{||\hat{\mathbf{x}}||}$
    $\mathbf{x} = \hat{\hat{\mathbf{x}}}$
else
    $\mathbf{x} = \hat{\mathbf{x}}$
end if
Compute inter target candidate similarity matrix: $\mathbf{Tar}_{\text{sim}}$, Equation (3-7)
Compute target candidate to background similarity matrix: $\mathbf{BG}_{\text{sim}}$, Equation (3-8)
for $k = 1$ to K **do**
    for $i = 1$ to N_l^+ **do**
        $\mathbf{J}_{\text{i}}$ = Compute Objective Function($\mathbf{Tar}_{\text{sim}}$(i,:), $\mathbf{BG}_{\text{sim}}$(i,:), $\mathbf{S}$), Equation (3-4)
    end for
    Extract signature with largest objective value: $\mathbf{s}_{\text{k}} = \arg\max_{\text{x}_{\text{i}}}(\mathbf{J}_{\text{i}})$
    $\mathbf{S} = \text{add}(\mathbf{s}_{\text{k}})$
end for
return Dictionary, $\mathbf{S}$

One method that is proposed, instead of trying every possible combination, initializes target signatures in a greedy manner. The method greedily selects the target candidate that maximizes the objective function until K target signatures have been initialized. It should be noticed that the objective function in Equation (3-4) includes a max operation across the target signatures. To compute the objective function at each of the K target iterations, the previously initialized target signatures are used to compute the objective function. The max operation is included to prioritize the target types in the positive bags that have not been included yet. This occurs because the target types that exists in bags that have not included

a target signature will increase the objective function. Furthermore, the objective function will not increase if the method selects the same initialized signature. This max operation encourages the algorithm to select target signatures that exists in all of the positive bags.

One drawback to this approach is that some target signatures are more like the background than others. If a target signature is more like the background, it is possible that it will actually provide a decrease in objective function if it is included. It has been noticed that even though this technique encourages learning all of the target types, if a target type is similar to the background, such that it will decrease the objective value, then the method will not include this target signature. The algorithm will avoid including a target signature if it causes a net decrease to the objective function.

Finally, it was noticed that there were operations that were used across multiple iterations of initializing a target. Namely, the similarity between the positive instances and similarity between the positive and negative instances was being computed during each of the K iterations. To exploit this recurrence and save computational costs, these values are stored in two matrices, $\mathbf{Tar}_{\text{sim}}$ and $\mathbf{BG}_{\text{sim}}$. The former stores the positive instance to positive instance similarities and the later stores the positive instance to negative instance similarities. Instead of recomputing the similarity of every target candidate to the other positive and negative instances, the algorithm can perform a matrix lookup to save computation time. These matrix computations are shown in Equation (3-7) and (3-8),

$$\mathbf{Tar}_{\text{Sim}}(i,j) = D(\mathbf{x}_{\text{i}}, \mathbf{x}_{\text{j}}), \quad \forall(\mathbf{x}_{\text{i}} \text{ and } \mathbf{x}_{\text{j}}) \in \forall\mathbf{B}^{+} \tag{3-7}$$

$$\mathbf{BG}_{\text{Sim}}(i,k) = D(\mathbf{x}_{\text{i}}, \mathbf{x}_{\text{k}}), \quad \forall\mathbf{x}_{\text{i}} \in \forall\mathbf{B}^{+} \text{ and } \forall\mathbf{x}_{\text{k}} \in \forall\mathbf{B}^{-} \tag{3-8}$$

where $D(\mathbf{x}_{\text{a}}, \mathbf{x}_{\text{b}})$ represents the similarity between $\mathbf{x}_{\text{a}}$ and $\mathbf{x}_{\text{b}}$, computed by ACE or SMF depending on the algorithm variation used. $\mathbf{Tar}_{\text{sim}}$ is a matrix that contains all of the similarity values between each positive instance. $\mathbf{BG}_{\text{sim}}$ is a matrix that contains all of the similarity values between each positive and negative instance. Due to the large number of instances in

real applications, it is difficult to store this many values in a single matrix, so these matrices are broken into smaller matrices, one for each bag.

3.3 Uniqueness Term Objective Function Initialization

The uniqueness term objective function initialization makes use of an additional term in the objective function to promote target uniqueness. It was noticed that certain target types are more like the background which it difficult to learn those signatures. Regardless of how many signatures were initialized, the penalty for including those target types caused by the second term of the objective function, outweighed the increase in objective function from the first term. The algorithm needed to be encouraged to select different target signatures even if it meant selecting signatures that were more like the background. Naturally, some target types are more like the background. To be detected, their corresponding signatures need to be included in the dictionary of signatures, $\mathbf{S}$. To accomplish this, a uniqueness term was added to the objective function to penalize the algorithm for initializing similar target signatures. The uniqueness term is simply the similarity between two target signatures and is shown in Equation (3-9).

$$U = \frac{\alpha}{\binom{K}{2}} \sum_{k=1}^{K-1} \sum_{l=k+1}^{K} D(\mathbf{s}_k, \mathbf{s}_l) \tag{3-9}$$

Where $\mathbf{s}_k$ and $\mathbf{s}_l$ are learned target concepts, K is the number of targets, and α is a hyperparameter to weight this term. Analyzing the uniqueness term, it can be seen that it is the average similarity between all of the combinations of target signatures, weighted by α. Equation (3-10) shows the modified objective function with the uniqueness term included. It can be seen that this term is subtracted in the objective function to provide a penalty to the algorithm if it initializes similar targets.

$$\arg\max_{\mathbf{s}} \frac{1}{N^+} \sum_{j:L_j=1} D(\mathbf{x}_j^*, \mathbf{s}) - \frac{1}{N^-} \sum_{j:L_j=0} \frac{1}{N_j^-} \sum_{\mathbf{x}_i \in B_j^-} D(\mathbf{x}_i, \mathbf{s}) - \frac{\alpha}{\binom{K}{2}} \sum_{k=1}^{K-1} \sum_{l=k+1}^{K} D(\mathbf{s}_k, \mathbf{s}_l) \tag{3-10}$$

Although the uniqueness term addresses the types of target signatures being initialized, it does not address how, or the order that these target signatures will be initialized. The issue of run time complexity stated in Section 3.2 still exists for this method, so the same strategy of greedily initializing target signatures is employed with this initialization.

3.4 Clustering Initialization Approaches

Additional initialization approaches using clustering methods have been investigated to determine if either performance or run time can be improved. By using clustering initialization approaches, the algorithm reduces computation time and obtains cluster centers that represent regions of the data. With this, the algorithm does not need to search through as many candidate points to initialize a target signature. Additionally, the initialization technique will learn a representation of the target signature that is representative of a subspace of the data, instead of initializing a single instance that is not guaranteed to represent a greater region of the target class.

3.4.1 K-Means

The first approach, referred to as K-Means, uses the K-Means clustering algorithm (MacQueen et al., 1967) to group all of the data, regardless of bag structure, into C clusters. Then, this initialization approach picks the cluster center that maximizes the MI-ACE objective function as the initialized target signature. This computation complexity is $O((N^+ + N^-)Ci + CN^+)$, where C is the number of clusters, N^+ and N^- are the number of positive and negative instances, respectively, and i is the number of iterations until K-Means converges. The first term corresponds to K-Means clustering, and the second term corresponds to determining the cluster centers that maximize the objective function. As long as the number of clusters, C, and the number of iterations i, remains small, the K-Means approach will have less of a computational cost than the original initialization method. This way the algorithm only needs to search through C candidates instead of N^+ candidates to initialize a target signature.

3.4.2 Ranked K-Means

The second approach, referred to as Ranked K-Means, uses the K-Means clustering algorithm (MacQueen et al., 1967) to create C clusters, regardless of bag structure. Instead of using the original objective function to score the cluster centers, a new multiple instance cluster rank is proposed to further reduce the computational cost. The multiple instance cluster rank of the k^{th} cluster, is the sum of the proportions of the elements in cluster k. The three terms of the rank are the proportion of positive bags that have an instance in cluster k, the proportion of instances in cluster k that came from a positive bag, and the proportion of instances in cluster k that came from a negative bag. This is formally defined in Equation (3-11).

$$Rank_{MIC}(k) = \frac{w^{B^+}\left(\frac{N_k^{B^+}}{N^{B^+}}\right) + w^{+}\left(\frac{N_k^{+}}{N^{+}}\right) - w^{-}\left(\frac{N_k^{-}}{N^{-}}\right) + w^{-}}{w^{B^+} + w^{+} + w^{-}}, \tag{3-11}$$

where N^{B^+}, N^+, N^- are the total number of positive bags, positive instances, and negative instances, respectively. $N_k^{B^+}$, N_k^+, N_k^- are the number of positive bags that have at least one instance in cluster k, the number of positive instances in cluster k, and the number of negative instances in cluster k, respectively. Finally, the weights w^{B^+}, w^+, and w^- are positive hyperparameter weights that are set based on the prior belief of the distributions of the constructed positive bags. Equation (3-11) adds w^- and is divided by the sum of the weights to force the rank to be from $[0, 1]$. If it is believed that the positive bags contain a majority of positive instances, then the first two weights should be higher than the last weight. Furthermore, if the positive bags contain a majority of negative instances, the last weight should dominate to require a minimal number of negative instances belonging to the cluster center that is initialized.

The computation complexity for this technique is $O((N^+ + N^-)Ci + C)$, where C is the number of clusters, N^+ and N^- are the number of positive and negative instances, respectively, and i is the number of iterations until K-Means converges. In this initialization technique, the

first term, corresponding to K-Means, dominates the complexity. The second term corresponds to determining the cluster center that maximizes the multiple instance cluster rank. Since all of the data proportions to compute the rank come straight from the clustering results, no additional computation complexity is needed to determine the cluster rank for each cluster. This is an indexed matrix lookup and therefore constant time, and thus not included in the second term of the computation complexity.

3.4.3 MI-CR

The MI-ClusterRegress algorithm (Wagstaff et al., 2008), shown in Section 2.3.1, clusters all of the data regardless of bag structure into C Gaussian distributions using a Gaussian Mixture Model (GMM) (Murphy, 2013). Then, an exemplar point is created in each positive bag for each of the C distributions. The exemplar point in each bag for the k^{th} distribution, is a weighted average of the instances in that bag, where each instance is weighted by the responsibility to the k^{th} distribution. In the original algorithm, these exemplar points are then used to train C separate regression models to allow each distribution to have their own local regression model.

This MI-ClusterRegress algorithm has been incorporated into this initialization method, referred to as MI-CR. In this initialization method, the clustering portion of MI-ClusterRegress, as well as creating the exemplar points, is used to reduce the number of instances that must be searched to initialize a target concept. Namely, $C \times N^{B^+}$ exemplar points are searched instead of all N^+ positive instances. Additionally, the created exemplar points are a combination of the instances in a positive bag, so the initialized exemplar point may be better at representing the variations in target representatives rather than using a single instance as the initialized target concept.

The computation complexity for MI-CR is $O((N^+ + N^-)Ci + CN^{B^+}N^+)$, where C is the number of clusters, N^+ and N^- are the number of positive and negative instances, respectively, N^{B^+} is the number of positive bags, and i is the number of iterations until Expectation Maximization for GMM converges. The first term corresponds to GMM's complexity. It can be

seen that GMM has an identical complexity as K-Means except the number of iterations, i, is dependent on the stopping condition threshold hyperparameter. GMM's complexity is largely affected by the stopping condition threshold. The second term corresponds to how many exemplar points are being considered as a potential target signature. For each bag, there are C exemplar points generated, so a total of $C \times N^{B^+}$ exemplar points are generated. Then the objective value is computed with complexity of N^+ for each of the exemplar points. This will dominate the computation time of MI-CR if there are many bags or many positive instances, but the computation time can also be largely affected by the stopping condition threshold used for GMM.

3.5 Original MI-ACE and MI-SMF Optimization Extended for Multiple Targets

The ideal objective function shown in Equation (3-4) is difficult to optimize. Instead the following objective was optimized and used during the experiments of this thesis, shown in Equation (3-12).

$$\max_{\mathbf{S}} \quad \frac{1}{N^+} \sum_{j:L_j=1} \sum_{\mathrm{s_k} \in \mathrm{S}} D(\mathbf{x}^*_{j,k}, \mathbf{s}_\mathrm{k}) - \frac{1}{N^-} \sum_{j:L_j=0} \frac{1}{N_j^-} \sum_{x_i \in B_j^-} \sum_{\mathrm{s_k} \in \mathrm{S}} D(\mathbf{x}_i, \mathbf{s}_k) \tag{3-12}$$

The optimization steps done in the original MI-ACE and MI-SMF algorithms, reviewed in Section 2.1.5, can be extended to update each target signature individually, shown in the update Equation (3-13). This optimization update equation can be applied to the initialized targets, but does not address the possibility of positive bags containing different target types. This is an issue because of how a target signature would be optimized. It can be seen in Equation (3-13) that the updated target signature is the average of the positive bag representatives minus the average of the background.

$$\hat{\hat{\mathbf{s}}}_k = \frac{\hat{\hat{\mathbf{t}}}}{\left\| \hat{\hat{\mathbf{t}}} \right\|} \text{ where } \hat{\hat{\mathbf{t}}} = \frac{1}{N^+} \sum_{j:L_j=1} \hat{\hat{\mathbf{x}}}^*_{j,k} - \frac{1}{N^-} \sum_{j:L_j=0} \frac{1}{N_j^-} \sum_{\hat{\hat{x}}_i \in B_j^-} \hat{\hat{\mathbf{x}}}_i \tag{3-13}$$

Since Equation (3-13) uses a positive bag representative, $\hat{\hat{\mathbf{x}}}^*_{j,k}$, from every j^{th} positive bag, the optimization will include bags that may not contain the target type being optimized.

This means that over iterations of the optimization process, the target signature will become an average of all of the positive bag representatives. This causes one of three possibilities to occur. One, the optimized signature would be an average of all or some of the target types. This is not desired. If the optimized signature is an average of the target concepts, this signature does not actually exist, and will likely not be useful for detecting any target types. Furthermore, it was observed that during optimization, the target signatures would often optimize to be the same target signature. Regardless of if the initialized target signatures were unique or not, the optimized signatures would converge to be the same target signature. This was occurring because of two reasons. First, the objective function in Equation (3-13) gives a higher objective function value for target signatures that are not like the background. Secondly, the objective function favors getting a high detection on as many positive bag representatives as possible. So the signature that all of the target representatives would optimize to would either be the one that was the most different than the background, or the target signature that existed the most across the positive bags.

The optimization update equation shown in Equation (3-13) was used to create the results in Chapter 4 because of the difficulties of optimizing the ideal update equation, shown in Equation (3-14). The ideal optimization update equation considers the maximum operation across the bag representatives, shown in the proposed multi-target objective function in Equation (3-4). The ideal optimization update equation is shown in Equation (3-14).

$$\hat{\mathbf{s}}_k = \frac{\hat{\mathbf{t}}}{\left\|\hat{\mathbf{t}}\right\|} \text{ where } \hat{\mathbf{t}} = \frac{1}{N^+} \sum_{\hat{\mathbf{x}}^* \in \mathbf{X}^*_{k,\text{sel}}} \hat{\mathbf{x}}^* - \frac{1}{N^-} \sum_{j:L_j=0} \frac{1}{N_j^-} \sum_{\hat{\mathbf{x}}_i \in B_j^-} \hat{\mathbf{x}}_i \tag{3-14}$$

$$\mathbf{X}^*_{k,\text{sel}} = \left\{ \hat{\mathbf{x}}^*_{j,k} : D\left(\hat{\mathbf{x}}^*_{j,k}, \hat{\mathbf{s}}_k\right) > D\left(\hat{\mathbf{x}}^*_{j,l}, \hat{\mathbf{s}}_l\right), \quad \forall l \neq k,\, j = 1, ..., N^+ \right\} \tag{3-15}$$

This shows that the signature being optimized, $\hat{\mathbf{s}}_k$, only uses the set of positive bag representatives that are selected, $\mathbf{X}^*_{k,\text{sel}}$. A positive bag representative, $\hat{\mathbf{x}}^*_{j,k}$, is selected to be included if it has a higher similarity to the target signature k, than any of the other similarities between the target signatures l, and their respective bag representatives, $\hat{\mathbf{x}}^*_{j,l}$. Intuitively, this

means that the signature being optimized will only include the positive bag representatives from bags that are believed to have the same target type as the signature being optimized. Due to the difficulties of using this ideal objective function, the originally extended objective function shown in Equation (3-13) was used.

In the Sections 3.6 and 3.7, two other optimization methods, the weighted optimization and the uniqueness term optimization, are proposed to mitigate the problems explained in Section 3.5. These methods attempt to optimize the target signatures using the correct, corresponding positive bag representatives during optimization by using the similarity to the target concepts being optimized.

3.6 Weighted Optimization Approach

The weighted optimization approach aims to mitigate the issue that arises when optimizing target representatives using multiple positive bags that contain different target types. Namely, it attempts to optimize the k^{th} target signature, based on the similarity of the target signature to it's corresponding positive bag representatives, $\mathbf{x}^*_{j,k}$. This similarity is then used as a weighting term, $w_{j,k}$, during optimization to more heavily weight the positive bag representatives that are similar to the target signature being optimized. The weight is defined below in Equation (3-16).

$$w_{j,k} = \frac{s^*_{j,k}}{\sum_{k=1}^{K} s^*_{j,k}} \tag{3-16}$$

where $s^*_{j,k}$ is the k^{th} target representative's similarity to it's corresponding bag representative, $\mathbf{x}^*_{j,k}$, in bag j. The bag representative is the sample from bag j with the largest similarity to the target signature and is defined in Equation (3-6) from Section 3.1. The similarity metric used is either ACE, Equation (2-23), or SMF, Equation (2-22), depending on if the multi-target MI-ACE or multi-target MI-SMF algorithm is being used, respectively. The update equation for the k^{th} target signature using the weighted optimization approach is shown in Equation (3-17).

$$\hat{\hat{\mathbf{s}}}_k = \frac{\hat{\hat{\mathbf{t}}}_k}{\left\|\hat{\hat{\mathbf{t}}}_k\right\|} \text{ where } \hat{\hat{\mathbf{t}}}_k = \frac{1}{N^+} \sum_{j:L_j=1} \frac{w_{j,k}\hat{\hat{\mathbf{x}}}^*_{j,k}}{\sum_{k=1}^{K} w_{j,k}} - \frac{1}{N^-} \sum_{j:L_j=0} \frac{1}{N_j^-} \sum_{\hat{\hat{x}}_i \in B_j^-} \hat{\hat{\mathbf{x}}}_i \tag{3-17}$$

Here it can be seen that, by optimizing the target signatures while using a weighted sum of positive bag representatives, the optimization process is able to update each target signature with an average of the positive bag representatives that it is the most alike. In turn, the positive bag representatives that are the most alike, are most likely target signatures that are of the same target type. Therefore each initialized target signature will be optimized with the positive bag representatives that are the same target type. Furthermore, the weighted update is normalized by the sum of the weights for all of the K target signatures. If there are other target signatures that are more like the positive bag representative, then the weight will be reduced. This was used to encourage the target representatives to be unique, because if there is another representative that is similar to the bag representative, then it is undesired for the target representative being optimized to use that target representative in it's update equation.

An optional Radial Basis Function (RBF) kernel is introduced to map the weights, which are the similarity between a target and a bag representative, towards 0 or 1 to create more crisp weights. The RBF kernel mapping is shown in Equation (3-18) and used in Line 12 of Algorithm 5.

$$w'_{j,k} = K(w_{j,k}, w_{k,max}) = \exp\left(- \frac{||w_{j,k} - w_{k,max}||^2}{2\sigma^2} \right) \tag{3-18}$$

$$w_{k,max} = \max(w_{j,k}) \forall j \tag{3-19}$$

When the RBF kernel is used, each weight, $w_{j,k}$, is replaced with its corresponding kernel mapping weight, $w'_{j,k}$. The kernel mapping is centered at the max weight value, $w_{k,max}$, where the max is taken across all of the weight values for a given k target signature, shown in Equation (3-19). In other words, the max weight is taken across all of the bags representatives, and will be the maximum similarity between the target signature and all of positive bag representatives. The kernel is centered at $w_{k,max}$ so that any other weight, $w_{j,k}$, that is close to the max weight will be mapped towards 1, while weights that are significantly lower will

be pushed towards 0. It can be seen that this will allow the optimization process to only optimize a target signature with positive bag representatives that are similar to it. Any bag representative that is not similar enough to the target signature, i.e. that bag does not have the target type being optimized, will have a very low weight during optimization. The bandwidth parameter, σ, controls how much the values get pushed towards 0 or 1. A smaller σ creates a more crisp mapping, while a larger σ creates a softer mapping, and the more equally distributed the weights will be.

Lastly, because the algorithm uses floating point weighted updates, the algorithm will never converge unless a stopping threshold is used. To determine when the algorithm is no longer making meaningful updates, an ACE similarity is used to determine if the signature is changing across iterations. If the similarity from the previous iteration to the current iteration is greater than the stopping threshold then the algorithm has fully optimized that signature, and the signature is no longer updated. This is shown in Equation (3-20).

$$D_{ACE}\left(\mathbf{s}_k^{(t)}, \mathbf{s}_k^{(t-1)}\right) = \left(\hat{\mathbf{s}}_k^{(t)\,T} \hat{\mathbf{s}}_k^{(t-1)}\right) > \theta_{stop} \tag{3-20}$$

where $\mathbf{s}_k^{(t)}$ is the current iteration of the k^{th} target signature, $\mathbf{s}_k^{(t-1)}$ is the previous iteration of the k^{th} target signature, and θ_{stop} is the stopping threshold that should be set very close, but slightly smaller than 1. The algorithm pseudocode is provided in Algorithm 5.

Algorithm 5 Multi-Target MI-SMF/MI-ACE Weighted Optimization

Compute $\boldsymbol{\mu}_{\mathrm{b}}$ and $\boldsymbol{\Sigma}_{\mathrm{b}}$
Subtract the background mean and whiten all instances, $\hat{\mathbf{x}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{U}^{T}(\mathbf{x} - \boldsymbol{\mu}_{\mathrm{b}})$
if MI-ACE **then**
    normalize: $\hat{\hat{\mathbf{x}}} = \frac{\hat{\mathbf{x}}}{||\hat{\mathbf{x}}||}$
end if
Initialize $\hat{\hat{\mathbf{s}}}$, as the instance in a positive bag that maximizes the obj. fun., Equation (3-10)
repeat
    for $k = 1$ to K **do**
        Update $\hat{\hat{\mathbf{x}}}^{*}_{\mathrm{j,k}}$, for $\hat{\hat{\mathbf{s}}}_{\mathrm{k}}$ in each positive bag, $\mathbf{B}^{+}_{\mathrm{j}}$, using Equation (3-6)
        Compute weights, $w_{j,k}$, using Equation (3-16)
        if RBF Kernel **then**
            Use mapped weights from Equation (3-18) $w_{j,k} = K(w_{j,k}, w_{k,max})$
        end if
        Update $\hat{\hat{\mathbf{s}}}_{k}$ using Equation (3-17)
    end for
until Stopping Criterion Reached
Normalize and De-whitened target signatures, $\mathbf{s}_{\mathrm{k}} = \frac{\mathbf{t}_{\mathrm{k}}}{||\mathbf{t}_{\mathrm{k}}||}$, where $\mathbf{t}_{\mathrm{k}} = \mathbf{U}\mathbf{D}^{-\frac{1}{2}}\hat{\hat{\mathbf{s}}}_{\mathrm{k}}$
return All $\mathbf{s}_{\mathrm{k}}$ as $\mathbf{S}$

3.7 Uniqueness Term Objective Function Optimization

The last optimization technique is using the modified objective function to promote target concept uniqueness during optimization. It was noticed that although an initialization method may be able to select target concepts that represent the various target types, during optimization using the extended original optimization in Equation (3-13), the target types would no longer remain unique, and would end up converging to be the same target concept. This is a problem in applications where target types are more similar than different. The extended original optimization technique in Section 3.5 updates each signature by averaging all

of the positive bag representatives. This can pose a problem though because if the initialized signatures are alike, they are likely to pick similar or even the same positive bag representatives. It was often seen that the optimized signatures would converge to be the same signature.

To mitigate this problem, a uniqueness term was added to the objective function, shown in Equation (3-21), as described in Section 3.3. To optimize the objective function with the uniqueness term, shown in Equation (3-22), the objective function can be posed as a lagrangian problem and the update equation for the k^{th} target signature can be solved for. The lagrangian is written in Equation (3-23). The detection statistic is expanded out showing the whitened data as well as the inner product. The derivation is demonstrated in Equation (3-24) - (3-26) using the ACE scenario but the same steps would be done for SMF. Finally, the update equation for the k^{th} target signature is solved for an written in Equation (3-27).

$$\max_{\mathbf{S}} \quad \frac{1}{N^+}\sum_{j:L_j=1}\max_{s_k\in \mathrm{S}}(D(\mathbf{x}^*_{j,k},\mathbf{s}_\mathrm{k}))-\frac{1}{N^-}\sum_{j:L_j=0}\frac{1}{N_j^-}\sum_{x_i\in B_j^-}\max_{s_k\in \mathrm{S}}(D(\mathbf{x}_i,\mathbf{s}_k))-\frac{\alpha}{\binom{K}{2}}\sum_{k=1}^{K-1}\sum_{l=k+1}^{K}D(\mathbf{s}_k,\mathbf{s}_l) \tag{3-21}$$

$$\max_{\mathbf{S}} \quad \frac{1}{N^+}\sum_{j:L_j=1}\max_{\hat{\mathbf{s}}_k\in \mathrm{S}}\left(\hat{\mathbf{x}}^{*T}_{j,k}\hat{\mathbf{s}}_\mathrm{k}\right)-\frac{1}{N^-}\sum_{j:L_j=0}\frac{1}{N_j^-}\sum_{\hat{x}_i\in B_j^-}\max_{\hat{\mathbf{s}}_k\in \mathrm{S}}\left(\hat{\mathbf{x}}_i^T\hat{\mathbf{s}}_k\right)-\frac{\alpha}{\binom{K}{2}}\sum_{k=1}^{K-1}\sum_{l=k+1}^{K}\left(\hat{\mathbf{s}}_k^T\hat{\mathbf{s}}_l\right) \tag{3-22}$$

$$\mathcal{L}=\frac{1}{N^+}\sum_{j:L_j=1}\max_{\hat{\mathbf{s}}_k\in \mathrm{S}}\left(\hat{\mathbf{x}}^{*T}_{j,k}\hat{\mathbf{s}}_\mathrm{k}\right)-\frac{1}{N^-}\sum_{j:L_j=0}\frac{1}{N_j^-}\sum_{\hat{x}_i\in B_j^-}\max_{\hat{\mathbf{s}}_k\in \mathrm{S}}\left(\hat{\mathbf{x}}_i^T\hat{\mathbf{s}}_k\right)-\frac{\alpha}{\binom{K}{2}}\sum_{k=1}^{K-1}\sum_{l=k+1}^{K}\left(\hat{\mathbf{s}}_k^T\hat{\mathbf{s}}_l\right)-\lambda\left(\hat{\mathbf{s}}_\mathrm{k}^\mathrm{T}\hat{\mathbf{s}}_\mathrm{k}-1\right) \tag{3-23}$$

$$\frac{\partial\mathcal{L}}{\partial\hat{\mathbf{s}}_\mathrm{k}}=\frac{1}{N^+}\sum_{j:L_j=1}\hat{\mathbf{x}}^*_{j,k}-\frac{1}{N^-}\sum_{j:L_j=0}\frac{1}{N_j^-}\sum_{\hat{x}_i\in B_j^-}\hat{\mathbf{x}}_i-\frac{\alpha}{\binom{K}{2}}\sum_{k=1}^{K-1}\sum_{l=k+1}^{K}\left(\hat{\mathbf{s}}_l\right)-2\lambda\hat{\mathbf{s}}_\mathrm{k} \tag{3-24}$$

$$\hat{\mathbf{s}}_\mathrm{k}=\frac{1}{2\lambda}\left(\frac{1}{N^+}\sum_{j:L_j=1}\hat{\mathbf{x}}^*_{j,k}-\frac{1}{N^-}\sum_{j:L_j=0}\frac{1}{N_j^-}\sum_{\hat{x}_i\in B_j^-}\hat{\mathbf{x}}_i-\frac{\alpha}{\binom{K}{2}}\sum_{k=1}^{K-1}\sum_{l=k+1}^{K}\hat{\mathbf{s}}_l\right) \tag{3-25}$$

$$\lambda = \frac{||\hat{\hat{\mathbf{t}}}||}{2} \tag{3-26}$$

$$\hat{\hat{\mathbf{s}}}_k = \frac{\hat{\hat{\mathbf{t}}}}{\left\|\hat{\hat{\mathbf{t}}}\right\|} \text{ where } \hat{\hat{\mathbf{t}}} = \frac{1}{N^+}\sum_{j:L_j=1}\hat{\hat{\mathbf{x}}}^*_{j,k} - \frac{1}{N^-}\sum_{j:L_j=0}\frac{1}{N_j^-}\sum_{\hat{\hat{\mathbf{x}}}_i\in B_j^-}\hat{\hat{\mathbf{x}}}_i - \frac{\alpha}{\binom{K}{2}}\sum_{k=1}^{K-1}\sum_{l=k+1}^{K}\hat{\hat{\mathbf{s}}}_l \tag{3-27}$$

The algorithm works very similarly to the original optimization shown in Algorithm 1 and the pseudocode for the optimization with the uniqueness term has been provided in Algorithm 6.

Algorithm 6 Multi-Target MI-SMF/MI-ACE with Uniqueness Term

Compute $\boldsymbol{\mu}_\mathrm{b}$ and $\boldsymbol{\Sigma}_\mathrm{b}$
Subtract the background mean and whiten all instances, $\hat{\mathbf{x}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{U}^T(\mathbf{x} - \boldsymbol{\mu}_\mathrm{b})$
if MI-ACE **then**
    normalize: $\hat{\hat{\mathbf{x}}} = \frac{\hat{\mathbf{x}}}{||\hat{\mathbf{x}}||}$
end if
Initialize $\hat{\hat{\mathbf{s}}}$, as the instance in a positive bag that maximizes the obj. fun., Equation (3-10)
repeat
    for $k = 1$ to K **do**
        Update $\mathbf{x}^*_{\mathrm{j,k}}$, for $\hat{\hat{\mathbf{s}}}_\mathrm{k}$ in each positive bag, $\mathbf{B}^+_\mathrm{j}$, using Equation (3-6)
        Compute average of other target signatures, $\hat{\hat{\mathbf{s}}}_l$, where $l \neq k$, at current iteration
        Update $\hat{\hat{\mathbf{s}}}_k$ using Equation (3-27)
    end for
until Stopping Criterion Reached
Normalize and De-whitened target signatures, $\mathbf{s}_\mathrm{k} = \frac{\mathrm{t_k}}{||\mathrm{t_k}||}$, where $\mathbf{t}_\mathrm{k} = \mathbf{U}\mathbf{D}^{-\frac{1}{2}}\hat{\hat{\mathbf{s}}}_\mathrm{k}$
return All $\mathbf{s}_\mathrm{k}$ as $\mathbf{S}$

CHAPTER 4
EXPERIMENTAL RESULTS

The Multi-Target MI-ACE and Multi-Target MI-SMF algorithms were applied to two datasets, a hyperspectral synthetic dataset and the MUUFL Gulfport hyperspectral dataset and was compared to other MIL algorithms. The receiver operating characteristic (ROC) curve was used to analyze the algorithm performances.

4.1 Synthetic Hyperspectral Target Detection Data

A synthetic hyperspectral data is needed to showcase how each proposed algorithm works in different, controlled test cases. Using synthetic data also provides preliminary results that will emphasize the technique of each algorithm over the difficulty of the dataset.

4.1.1 Data Generation

The synthetic data is generated using the technique outlined in (Jiao and Zare, 2015) which uses four hyperspectral signatures from the ASTER library (Baldridge et al., 2009) to generate positive and negative bags. The source code used was from the github repository by Jiao and Zare (2016). The four spectra used to generate data are the Red Slate, Verde Antique, Phyllite, and Pyroxenite from the rock class. The hyperspectral signatures contain 211 bands ranging in wavelength from 0.4μm to 2.5μm. These four spectra are shown in Figure 4-1.

To generate the data, each i^{th} data point in bag j, $\mathbf{x}_{ij}$, is generated as a linear combination of the four endmember spectra shown in Figure 4-1. A target datapoint is generated using Equation (4-1).

$$\mathbf{x}_{ij} = \sum_{k=1}^{K} a_{ik}\mathbf{d}_k^+ + \sum_{m=1}^{M} a_{im}\mathbf{d}_m^- + \varepsilon_{ij}, \quad \text{s.t.} \quad \exists a_{ik} \neq 0 \quad \text{and} \quad \sum_{k=1}^{K} a_{ik} + \sum_{m=1}^{M} a_{im} = 1 \tag{4-1}$$

Here it can be seen that the data point is a sum of endmembers and a noise term, ε_{ij}. Specifically, a target data point is a weighted sum of K target endmembers, $\mathbf{d}_k^+$, and M background endmembers, $\mathbf{d}_m^-$, weighted by a proportion value, a_{ik} for the target endmembers,

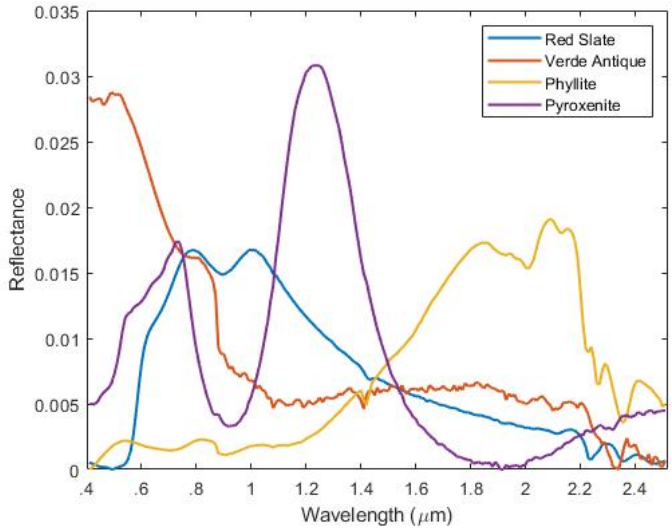


Figure 4-1. The four hyperspectral signatures used to generate synthetic data.

and a_{im} for the background endmembers. It can be seen that a positive data point must have some target endmember proportion not equal 0 by definition of what a target datapoint is. To generate a truly background datapoint Equation (4-2) is introduced.

$$\mathbf{x}_{\mathrm{ij}} = \sum_{m=1}^{M} a_{im}\mathbf{d}_{\mathrm{m}}^{-} + \varepsilon_{ij}, \quad \text{s.t.} \quad \exists a_{im} \neq 0 \quad \text{and} \quad \sum_{m=1}^{M} a_{im} = 1 \tag{4-2}$$

Here, the i^{th} data point, $\mathbf{x}_{\mathrm{ij}}$, in bag j is modeled as a sum of background endmembers, $\mathbf{d}_{\mathrm{m}}^{-}$, and a noise term, ε_{ij}. It can be seen that a true background data point will only contain endmember proportions from the background class which is what a background data point is by definition.

The number of endmembers for each data point are drawn randomly. The proportion values, a_{ik} and a_{im}, in Equation (4-1) and (4-2) are drawn from a Dirichlet distribution controlled by the $\boldsymbol{\alpha}_{\mathrm{mean}}$ parameter. The $\boldsymbol{\alpha}_{\mathrm{mean}}$ parameter is the expected mean value for the generating process with varying levels of magnitude controlled by the magnitude of $\boldsymbol{\alpha}_{\mathrm{mean}}$. The Dirichlet distribution is used as it provides proportion values that are non zero and sum to 1. For more details on how the data is generated see Algorithms 7 and 8.

Algorithm 7 shows the technique used to generate MIL positive and negative bags given a dictionary of target signatures, $\mathbf{D}^+$, and a dictionary of background signatures, $\mathbf{D}^-$. Algorithm 8 shows the process of generating data points, $\mathbf{x}$, based on a bag-level and instance-level label. In the pseudocode the variables are as follows: K^+ is the desired number of positive bags, K^- is the desired number of negative bags, N_i is the number of instances in each bag, N_{tar} is the number of target signatures in each positive bag, N_b is the minimum number of background signatures used to generate each data point, $\boldsymbol{\alpha}_{\text{mean}}$ is the desired mean target proportion value, and σ is a parameter to control the variance of the proportions. The algorithms return a fully synthetic data matrix, $\mathbf{X}$, consisting of individual data vectors, $\mathbf{x}$, along with their binary bag-level labels, L, and their binary instance-level labels, I.

Algorithm 7 Pseudo Code for Generating Synthetic Data as Bags

Inputs: $\mathbf{D}^+$, $\mathbf{D}^-$, K^+, K^-, N_i, N_{tar}, N_b, $\boldsymbol{\alpha}_{\text{t\_mean}}$, σ
Outputs: $\mathbf{X} = \bigcup \mathbf{x}_{\text{ij}}$, $\mathbf{L}$, $\mathbf{I}$

for <$i = 1$ to K^+> **do**
    $\mathbf{L}(i) = 1$
    for $j = 1$ to N_{tar} **do**
        $\mathbf{I}(i,j) = 1$
        $\mathbf{x}_{\text{i,j}}$ = Algorithm 8 with parameters $\{\mathbf{D}^+, \mathbf{D}^-, \mathbf{L}(i), \mathbf{I}(i,j), N_b, \boldsymbol{\alpha}_{\text{t\_mean}}, \text{ and } \sigma\}$
    end for
    for $j = N_{tar} + 1$ to N_i **do**
        $\mathbf{I}(i,j) = 0$
        $\mathbf{x}_{\text{i,j}}$ = Algorithm 8 with parameters $\{\mathbf{D}^+, \mathbf{D}^-, \mathbf{L}(i), \mathbf{I}(i,j), N_b, \boldsymbol{\alpha}_{\text{t\_mean}}, \text{ and } \sigma\}$
    end for
end for
for $i = K^+ + 1$ to $K^+ + K^-$ **do**
    $\mathbf{L}(i) = 0$
    for <$j = 1$ to N_i> **do**
        $\mathbf{I}(i,j) = 0$
        $\mathbf{x}_{\text{i,j}}$ = Algorithm 8 with parameters $\{\mathbf{D}^+, \mathbf{D}^-, \mathbf{L}(i), \mathbf{I}(i,j), N_b, \boldsymbol{\alpha}_{\text{t\_mean}}, \text{ and } \sigma\}$
    end for
end for

Algorithm 8 Pseudo Code for Generating Linearly Mixed Data Points Given Bag-level and Point-level Label

Inputs: $\mathbf{D}^+$, $\mathbf{D}^-$, L_i, $l_{i,j}$, N_b, $\boldsymbol{\alpha}_{\text{t\_mean}}$, and σ

Outputs: $\mathbf{x}$

if $L_i \& l_{ij}$ **then**
  Draw a random integer k between $[1, K]$
  Randomly select k dictionary elements, $\mathbf{D}_{\text{k}}^+$, from $\mathbf{D}^+$
  Draw a random integer m between $[N_b, M]$
  Randomly select m dictionary elements, $\mathbf{D}_{\text{m}}^-$, from $\mathbf{D}^-$
  if $m == 0$ **then**
    $\boldsymbol{\alpha}_{\text{mean}} = \boldsymbol{\alpha}_{\text{t\_mean}}$
  else
    $\boldsymbol{\alpha}_{\text{mean}} = \sigma \cdot [\boldsymbol{\alpha}_{\text{t\_mean}}, \frac{1-\boldsymbol{\alpha}_{\text{t\_mean}}}{m} \times \mathbf{1}_{1\times m}]$
  end if
  $\mathbf{a} \leftarrow$ sample $k + m$ random values from Dirichlet distribution using $\boldsymbol{\alpha}_{\text{mean}}$
  Generate $\mathbf{x}$ using Equation (4-1) with $\mathbf{a}$ and $[\mathbf{D}_{\text{k}}^+, \mathbf{D}_{\text{m}}^-]$
else
  Uniformly draw integer m between $[\max(1, N_b), M]$
  $\boldsymbol{\alpha}_{\text{mean}} = \sigma \cdot \mathbf{1}_{1\times m}$
  Randomly select m dictionary elements, $\mathbf{D}_{\text{m}}^-$, from $\mathbf{D}^-$
  $\mathbf{a} \leftarrow$ sample m random values from Dirichlet distribution using $\boldsymbol{\alpha}_{\text{mean}}$
  Generate $\mathbf{x}$ using Equation (4-2) with $\mathbf{a}$ and $\mathbf{D}_{\text{m}}^-$
end if

4.1.2 Synthetic Data Experiments

In the following experiments the positive bags contain two target types to determine if the proposed algorithms can learn multiple target representations. The Phyllite and Pyroxenite signatures are used as targets while the other two signatures, Red Slate and Verde Antique, are used to generate the background samples. Unless otherwise stated, the experiments use the following parameters to generate training data. The number of positive bags $(K^+) = 25$, the number of negative bags $(K^-) = 25$, the number of points per bag $(N_i) = 20$, the number of target points per positive bag $(N_{tar}) = 4$, the expected target proportion per positive instance $(\boldsymbol{\alpha}_{\text{mean}}) = .2$, the variance $(\sigma) = 3$, and a desired Signal Noise Ratio (SNR) of 40 dB. The generated test data has a total of 25,000 true positive instances and 25,000 true negative instances. With a expected target proportion value $(\boldsymbol{\alpha}_{\text{mean}}) = .1$, the variance $(\sigma) = 10$, and a desired SNR of 30 dB.

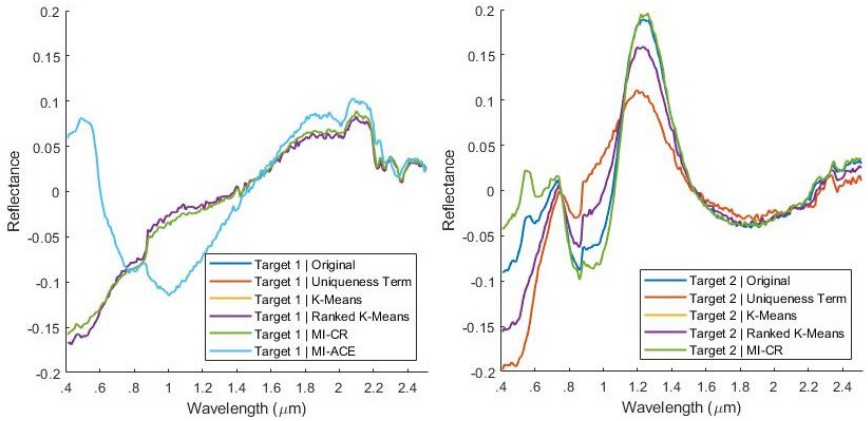
In Appendix A, experiments are shown to demonstrate how each version of the multi-target algorithm performs with various algorithm hyperparameter settings. The optimal settings from these experiments are then used to compare the proposed methods shown in Figures 4-2 - 4-9. In these experiments, the proposed initialization methods and proposed optimization methods are shown for both ACE and SMF. To test the optimization methods, the initialization method is held fixed for these experiments, using the original initialization method, so that the focus can be on the optimization methods. Additionally, the proposed algorithms are compared to their single target counterpart, MI-ACE or MI-SMF.

The learned target signatures for the multi-target MI-ACE and multi-target MI-SMF were used for detection with the ACE and SMF similarity statistic, respectively. The maximum ACE or SMF value, for all of the learned target signatures, was taken to be the confidence of an unknown sample being a target. Due to the background mean subtraction done during whitening, the proposed algorithms learned target concepts are not like the true target concepts. The learned target signatures are dewhitened, but the mean subtraction is not simple to add back in and so the concepts are compared with the mean background subtraction. With this, it is difficult to verify that the algorithms learn the "right" signatures by comparing them to the two, true target signatures directly, found in Figure 4-1. However, they can be verified by considering their general shape and how the background mean is expected to affect the resulting learned signatures. The ROC curve performance, as well as verifying that two different target signatures are learned, will additionally justify that the algorithms are learning the "right" signatures.

The run times in the following experiments were computed using an Intel Xeon CPU E5-1650 with 6 cores at a frequency of 3.60GHz on a machine running Windows 10 with 64.0 GB of RAM. The run time was determined by wrapping the desired part of the initialization code with the tic and toc function in Matlab.

The experiments in Figures 4-2 - 4-3 show the results of testing the proposed initialization approaches with ACE. The learned target signatures and their respective ROC curve

performance using the optimal parameters found in Appendix A are displayed. It can be seen that all of the proposed ACE methods initialize the right two signatures. Additionally, the single target MI-ACE learns one signature that is unable to detect both target signatures in the test set.



(a) (b)

Figure 4-2. Initialized signatures for different proposed initialization methods using (ACE) statistic with optimal parameters. (a) Estimated target signature 1. (b) Estimated target signature 2.

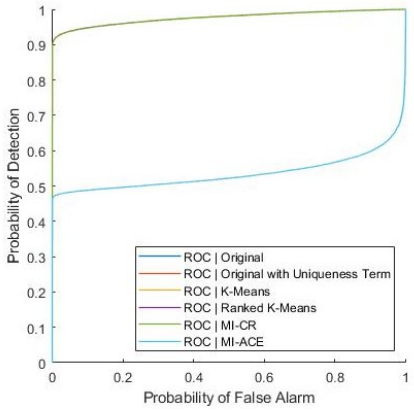
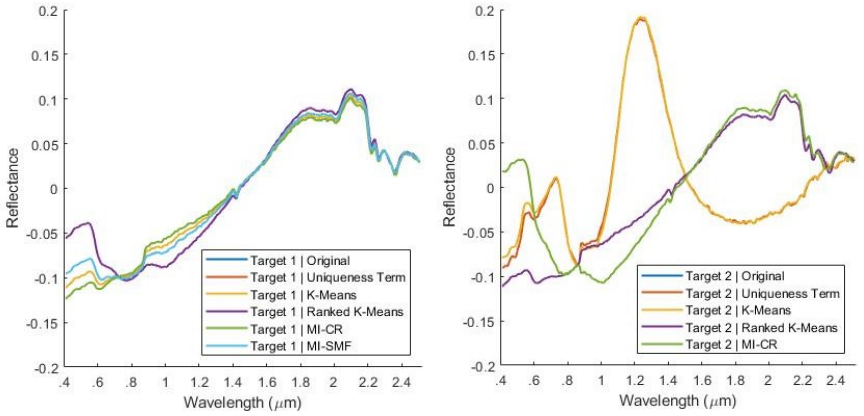


Figure 4-3. ROC curves for initialized signatures for different proposed initialization methods using (ACE) statistic with optimal parameters

The experiments in Figures 4-4 - 4-5 show the results of testing the proposed initialization approaches with SMF. The learned target signatures and their respective ROC curve

performance using the optimal parameters found in Appendix A are displayed. It can be seen that all of the proposed SMF methods initialize the right two signatures. Additionally, the single target MI-SMF learns one signature that is unable to detect both target signatures in the test set.



(a) (b)

Figure 4-4. Initialized signatures for different proposed initialization methods using (SMF) statistic with optimal parameters. (a) Estimated target signature 1. (b) Estimated target signature 2.

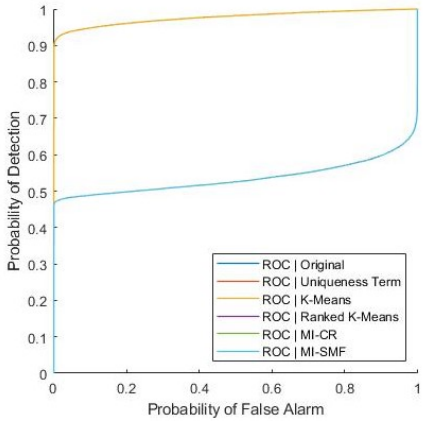


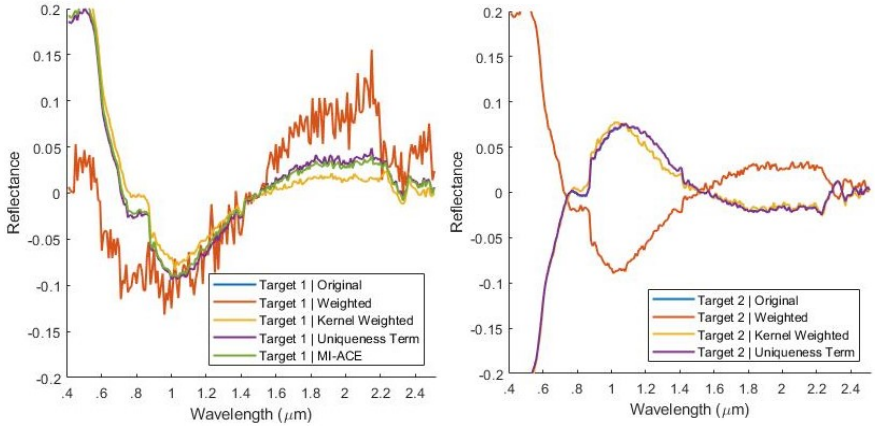
Figure 4-5. ROC curves for initialized signatures for different proposed initialization methods using (SMF) statistic with optimal parameters

Table 4-1. Area Under the Curve (AUC) and run times for each of the proposed initialization methods and the single target version of MI-ACE/SMF on the synthetic dataset

Method	(ACE) Time	(ACE) AUC	(SMF) Time	(SMF) AUC
Original	6.56	0.976	6.40	0.976
Uniqueness term	6.21	0.976	6.20	0.976
K-Means	0.178	0.976	0.162	0.976
Ranked K-Means	0.170	0.976	0.113	0.535
MI-CR	1.56	0.976	0.648	0.535
MI-ACE/SMF	4.21	0.535	4.21	0.535

It can be seen, that many of the performances are identical and are on top of each other in the ROC curves. This can be verified in Table 4-1 by seeing which algorithms have the same Area Under the Curve (AUC). The ACE results show that all of the proposed algorithms are able to initialize the right target signatures, while the single target MI-ACE is unable to learn a target signature that can detect both target types. The SMF results show that the original greedy initialization method, the uniqueness term method, and the K-Means method are able to initialize the right target signatures, while the other proposed algorithms initialize the same target signature twice. The single target MI-SMF algorithm is unable to learn a single target that can detect both target signatures.

The experiments in Figures 4-6 - 4-7 show the results of testing the proposed optimization approaches with ACE. The learned target signatures and their respective ROC curve performance using the optimal parameters found in Appendix A are displayed. It can be seen that only the original greedy optimization and the uniqueness term optimization optimize the signatures to be the right two signatures. The other proposed optimization approaches optimize the signatures to be unable to detect both target signatures in the test set.



(a) (b)

Figure 4-6. Optimized target signatures with the various proposed optimization techniques using (ACE) statistic. Initialized signatures are from K-Means method. (a) Estimated target signature 1. (b) Estimated target signature 2.

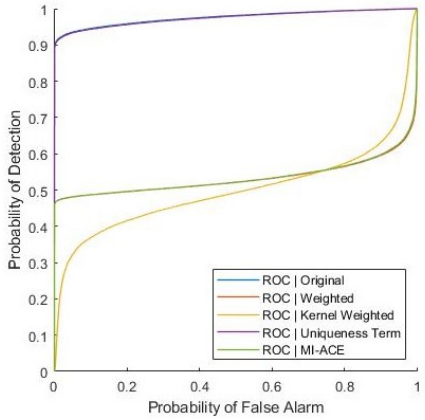
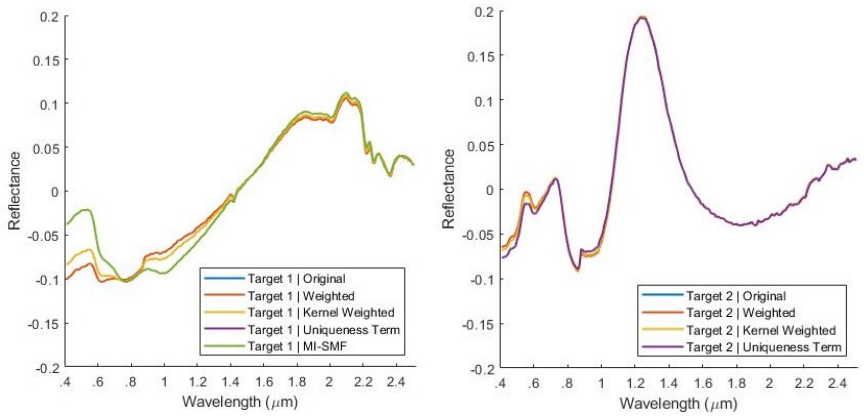


Figure 4-7. Optimized target signatures ROC curves with various proposed optimization techniques using (ACE) statistic. Initialized signatures are from K-Means method.

The experiments in Figures 4-8 - 4-9 show the results of testing the proposed optimization approaches with SMF. The learned target signatures and their respective ROC curve performance using the optimal parameters found in Appendix A are displayed. It can be seen that all of the proposed optimization methods are able to optimize the signatures to be

the right two signatures. The single target MI-SMF is only able to optimize a single target which cannot detect both target signatures in the test set.



(a) (b)

Figure 4-8. Optimized target signatures with the various proposed optimization techniques using (SMF) statistic. Initialized signatures are from K-Means method. (a) Estimated target signature 1. (b) Estimated target signature 2.

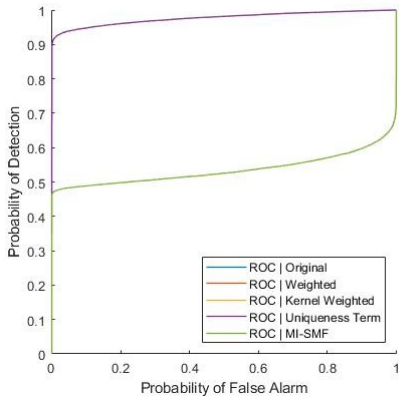


Figure 4-9. Optimized target signatures ROC curves with various proposed optimization techniques using (SMF) statistic. Initialized signatures are from K-Means method.

It can be seen, that many of the performances are identical and are on top of each other in the ROC curves. This can be verified in Table 4-2 by seeing which algorithms have the same Area Under the Curve (AUC). The ACE results show that only the original and uniqueness

Table 4-2. Area Under the Curve (AUC) for each of the proposed optimization methods and the single target version of MI-ACE/SMF on the synthetic dataset

Method	(ACE) AUC	(SMF) AUC
Original	0.975	0.976
Weighted	0.533	0.976
Kernel weighted	0.500	0.976
Uniqueness term	0.974	0.976
MI-ACE/SMF	0.534	0.535

term optimization methods are able to optimize the right target signatures, while all of the others are unable to learn a set of target signatures that can detect both target types. The single target MI-ACE algorithm is unable to learn a single target signature that is able to detect both target signatures in the test set. The SMF results show that all of the proposed optimization methods are able to optimize the signatures to be the right target signatures. The single target MI-SMF algorithm is unable to learn a single target that can detect both target signatures in the test set.

4.2 MUUFL Gulfport Hyperspectral Target Detection Data

The MUUFL Gulfport Hyperspectral data set (Gader et al., 2013) is chosen to test the proposed algorithms on a very challenging dataset. The dataset was collected over the University of Southern Mississippi-Gulfport campus in Long Beach, MS. The original dataset contains a total of 64 handmade targets that were constructed using wood frames and covered with 100% cotton, cloth panels. 57 of the handmade targets were selected to be included in the experiments that follow. The 57 targets were constructed to have different sizes of $0.5m$ x $0.5m$, $1m$ x $1m$, and $3m$ x $3m$. Four of the seven targets not included are large targets measuring 6m x 10m and were collected for the purposes of calibration. The other three of seven targets that are not included are the three Vineyard Green targets that were constructed using a different material than the Faux Vineyard Green targets. The 57 targets that are used contain 15 Brown, 15 Dark Green, 12 Faux Vineyard Green, and 15 Pea Green targets.

The data set contains a total of 325 x 337 pixels with 72 spectral bands corresponding to $375nm$ to $1050nm$ at a sampling interval of $10nm$. The ITRES CASI-1500 hyperspectral

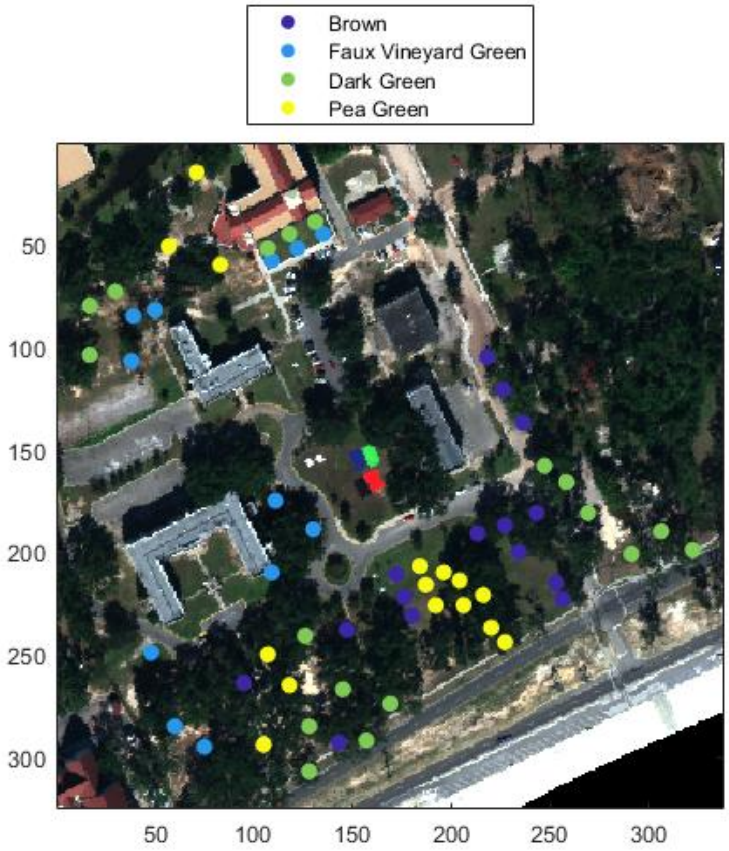


Figure 4-10. Scatter plot of the four types of target ground truth locations over the MUUFL Gulfport collection site RGB image

imager was used for the collection. The first four and last four spectral bands from the hyperspectral cube were found to be noisy and were removed during the experiments. The spatial resolution of each pixel is $1m^2$. Two different sets of the data, Gulfport Campus Flight one and Gulfport Campus Flight three, collected on November 8^{th} 2010, were used for algorithm cross validation training and testing in the experiments that follow. Gulfport Campus Flight one was used for training and Gulfport Campus Flight three was used for testing. These two flights were at the same altitude, 3500 feet, so they have the same spatial resolution which makes it possible to use the two for training and testing. The 57 targets are displayed over an an RGB image of the data collection in Figure 4-10.

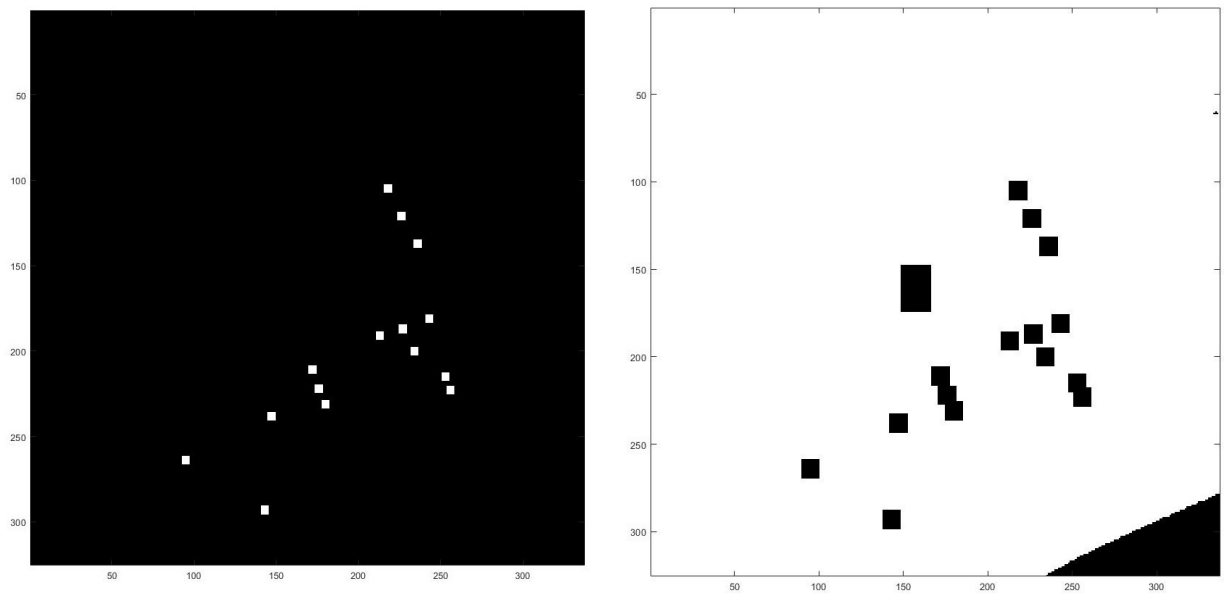
4.2.1 Individual Target Type Detection Experiments

For the experiments that follow, each individual target type is treated as the target class and the positive bags for training only contain the individual target type, i.e. for the Brown test case, only Brown targets were used during training. The "Truth" target signature for each target type was manually extracted from the image to provide a performance comparison of the ground truth target signature versus a learned target signature from the proposed algorithms. The cosine similarity (normalized inner product) between the "Truth" signature and a test pixel is used to generate a confidence for every test pixel. When computing the "Truth" signature confidence for the SMF experiments, the cosine similarity is not normalized by the test pixel, similar to SMF. Since there is only one target type in these experiments, a single signature is learned for the proposed methods to showcase the different initialization methods. The ACE or SMF detection statistic is used with the corresponding proposed ACE or SMF algorithms to determine the confidence of a test sample being a target.

The resulting confidences are scored using the bullwinkle scoring method outlined by Glenn et al. (2013) with a halo size of 2, which accounts for the inaccuracies of the recording GPS system in the test data. The bullwinkle scoring method takes the maximum value within a region around the ground truth, known as a halo, and uses the maximum value from the halo to score the results. Similarly to the synthetic dataset, due to the background mean subtraction done during whitening, the proposed algorithms' learned target concepts are not like the true target concepts. The learned target signatures are dewhitened, but the mean subtraction is not simple to add back in, so the concepts are compared with the mean background subtracted. With this, it is difficult to verify that the algorithms learn the "right" signatures by comparing them directly to the ground truth signatures. However, they can be verified by considering their general shape and how the background mean is expected to affect the resulting learned signatures. The ROC curve performance, as well as comparing the target signatures, will justify that the algorithms are learning the "right" signatures. The Normalized

Area Under the Curve (NAUC) is computed within the window of 0 to 0.001 FAR/m^2 of the ROC.

To create each positive bag for training, a pixel window of size $5x5$ around the corresponding GPS ground truth location was used for each target of interest. This window size is chosen to account for the GPS inaccuracy used for recording the ground truth. Then most of the remaining area is used as one large negative bag. An additional two pixel wide keep out region is added around each of the positive bags, as well as a mask is added to remove the bottom right corner which is not valid data in the original image. Lastly, the four large atmospheric correction targets in the middle of the image are masked out. The remaining region in white corresponds to the large negative bag used for training. The bagging process for the "Brown" targets is showcased in Figure 4-11. Following, the number of targets and size of each individual target type is shown in Table 4-3.



(a) (b)

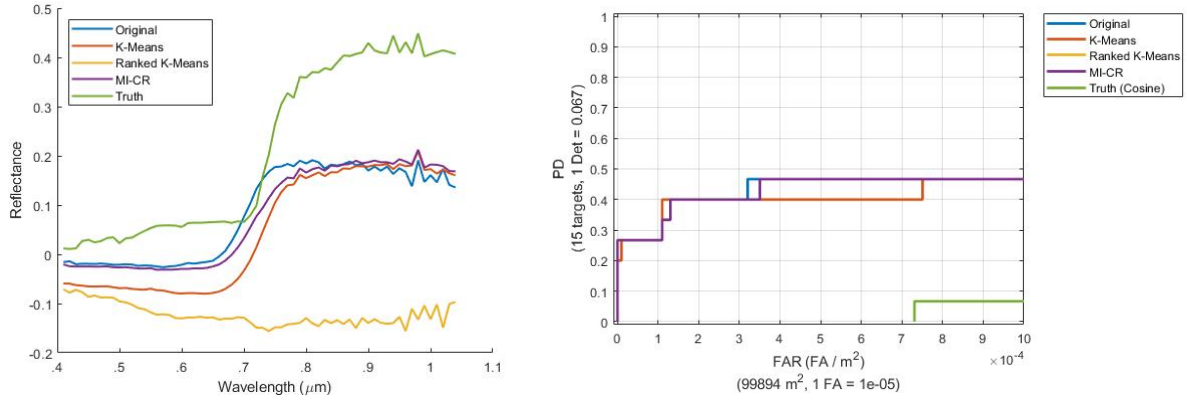
Figure 4-11. Bagging image masks for the MUUFL Gulfport dataset brown targets with a 5 x 5 window. The white region corresponds to the regions extracted during the bagging process. (a) Positive bag regions. (b) Negative bag region.

Table 4-3. The number of targets in each of the MUUFL Gulfport hyperspectral datasets for the single target experiments.

Target type	Size: $0.5m$ x $0.5m$	Size: $1m$ x $1m$	Size: $3m$ x $3m$	All sizes
Brown	5	5	5	15
Faux vineyard green	4	4	4	12
Dark green	5	5	5	15
Pea green	5	5	5	15

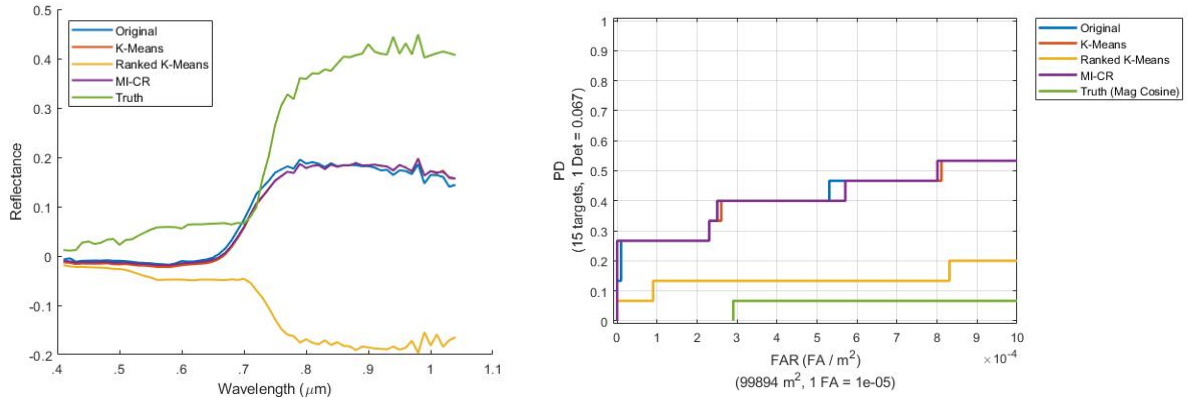
The following four Sections 4.2.1.1 - 4.2.1.4 show the results of initializing a single signature using the different proposed methods while training on an individual target type, i.e. Brown, Faux Vineyard Green, Dark Green, or Pea Green. The performances are compared using a ROC curve and the corresponding NAUC. The run times of the initialization methods are compared using the tic and toc functions from matlab. The run times in the following experiments were computed using an Intel Xeon CPU E5-1650 with 6 cores at a frequency of 3.60GHz on a machine running Windows 10 with 64.0 GB of RAM. The run time was determined by wrapping the desired part of the initialization code with the tic and toc function in Matlab.

4.2.1.1 Brown targets



(a) (b)

Figure 4-12. Experiment results for Multi-Target MI-ACE using different initialization techniques, on MUUFL Gulfport data, brown targets. (a) Initialized brown target signatures. (b) Brown targets ROC curve.



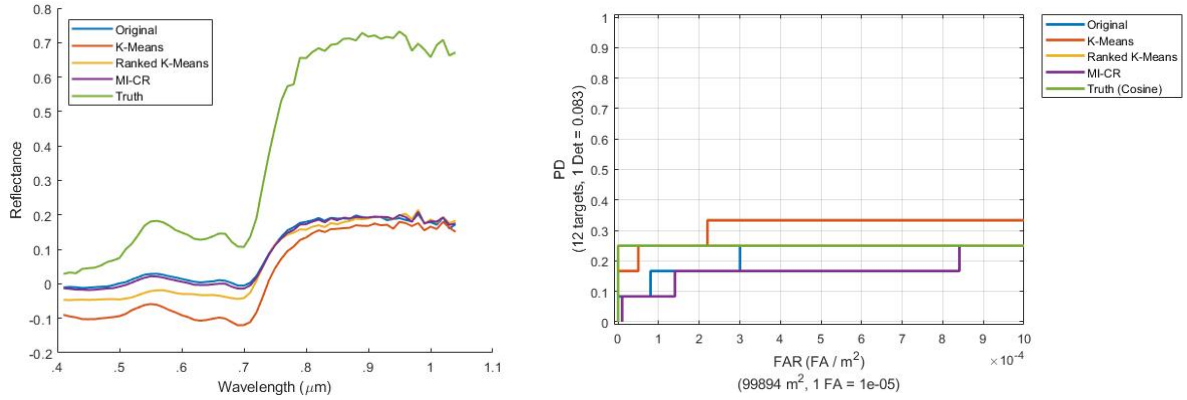
(a) (b)

Figure 4-13. Experiment results for Multi-Target MI-SMF using different initialization techniques, on MUUFL Gulfport data, brown targets. (a) Initialized brown target signatures. (b) Brown targets ROC curve.

Table 4-4. Brown target initialization experiments showing the proposed methods for ACE, SMF, and the ground truth. The run times and NAUC performances on the MUUFL Gulfport hyperspectral dataset are listed. MI-HE results obtained from (Jiao, 2017).

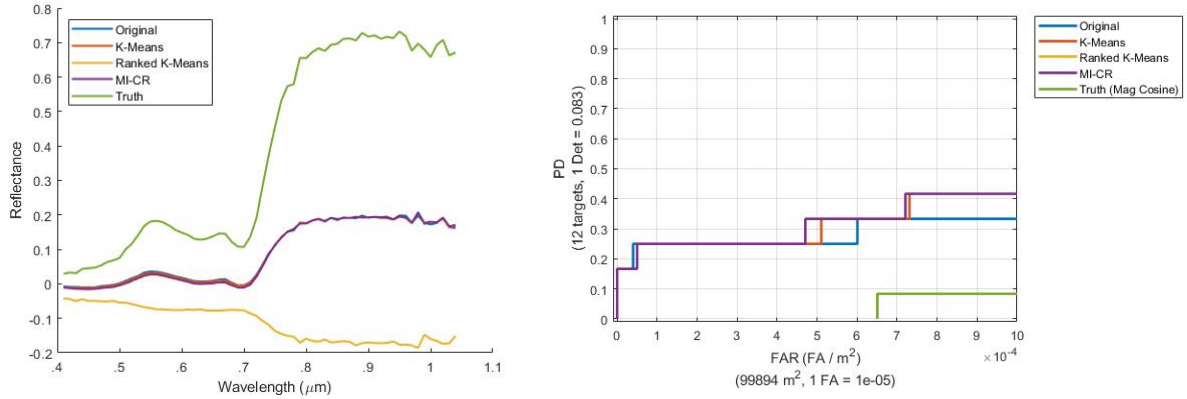
Method	(ACE) Time	(ACE) AUC	(SMF) Time	(SMF) AUC	(HSD) NAUC
Original	0.174	0.430	0.175	0.410	-
K-Means	0.013	0.401	0.015	0.408	-
Ranked K-Means	0.035	0	0.037	0.138	-
MI-CR	0.173	0.427	0.188	0.410	-
MI-HE	-	0.433	-	-	0.499
Truth	0	0.017	0	0.047	-

4.2.1.2 Faux vineyard green targets



(a) (b)

Figure 4-14. Experiment results for Multi-Target MI-ACE using different initialization techniques, on MUUFL Gulfport data, faux vineyard green targets. (a) Initialized faux vineyard green target signatures. (b) Faux vineyard green targets ROC curve.



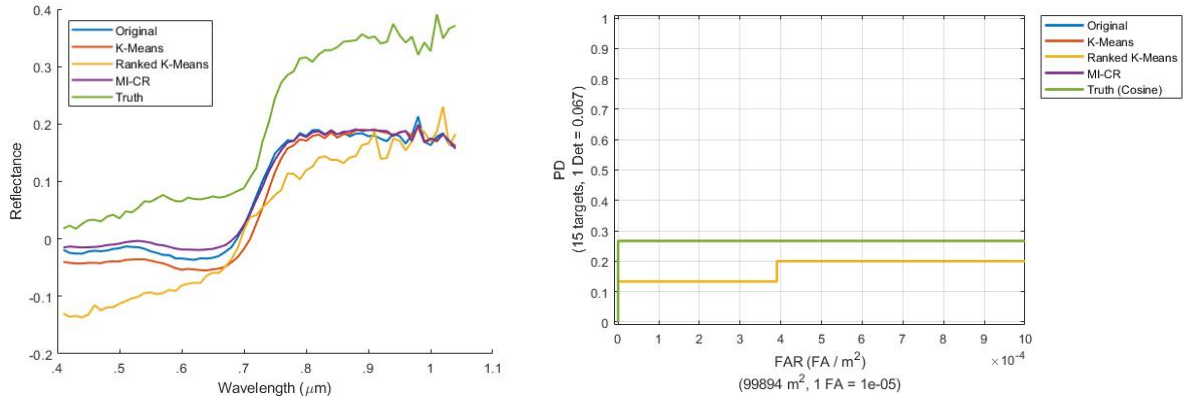
(a) (b)

Figure 4-15. Experiment results for Multi-Target MI-SMF using different initialization techniques, on MUUFL Gulfport data, faux vineyard green targets. (a) Initialized faux vineyard green target signatures. (b) Faux vineyard green targets ROC curve.

Table 4-5. Faux vineyard green target initialization experiments showing the proposed methods for ACE, SMF, and the ground truth. The run times and NAUC performances on the MUUFL Gulfport hyperspectral dataset are listed. MI-HE results obtained from (Jiao, 2017).

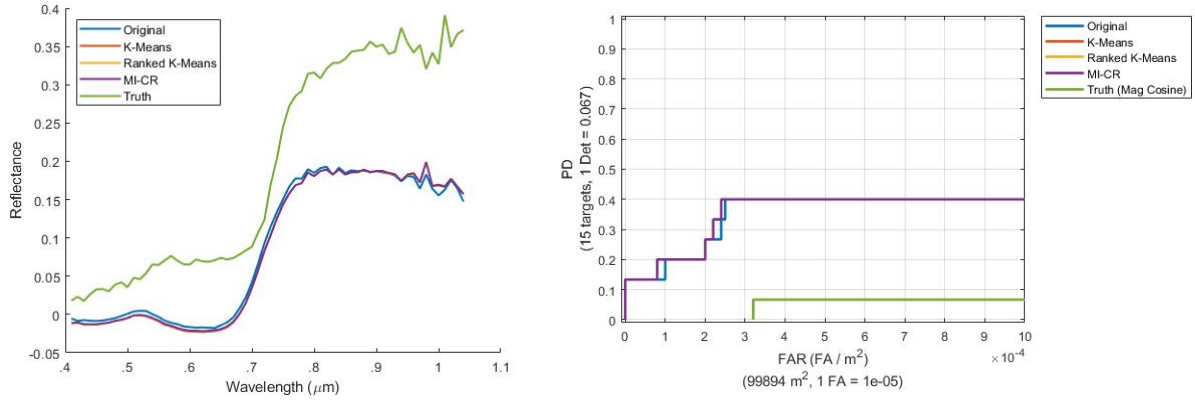
Method	(ACE) Time	(ACE) AUC	(SMF) Time	(SMF) AUC	(HSD) NAUC
Original	0.284	0.218	0.223	0.279	-
K-Means	0.017	0.310	0.017	0.309	-
Ranked K-Means	0.027	0	0.050	0	-
MI-CR	0.427	0.167	0.160	0.313	-
MI-HE	-	0.104	-	-	0.655
Truth	0	0.250	0	0.029	-

4.2.1.3 Dark green targets



(a) (b)

Figure 4-16. Experiment results for Multi-Target MI-ACE using different initialization techniques, on MUUFL Gulfport data, dark green targets. (a) Initialized dark green target signatures. (b) Dark green targets ROC curve.



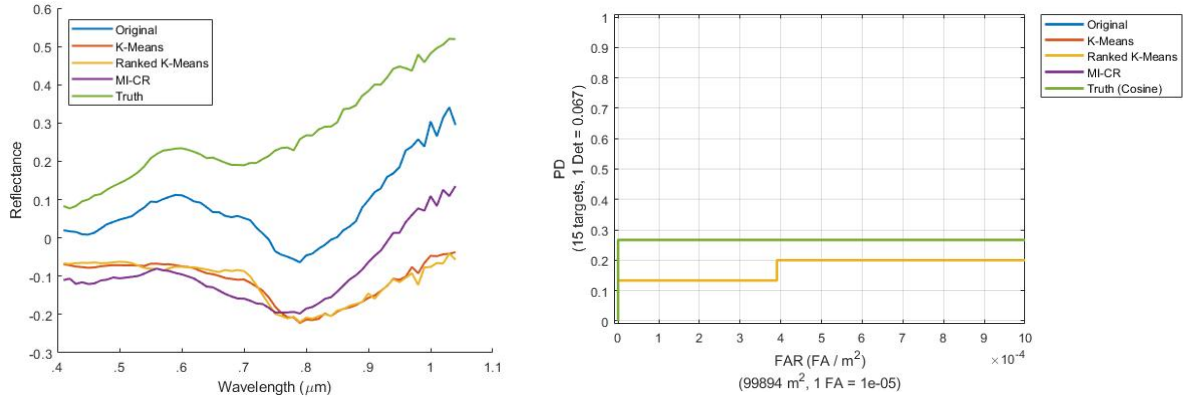
(a) (b)

Figure 4-17. Experiment results for Multi-Target MI-SMF using different initialization techniques, on MUUFL Gulfport data, dark green targets. (a) Initialized dark green target signatures. (b) Dark green targets ROC curve.

Table 4-6. Dark green target initialization experiments showing the proposed methods for ACE, SMF, and the ground truth. The run times and NAUC performances on the MUUFL Gulfport hyperspectral dataset are listed. MI-HE results obtained from (Jiao, 2017).

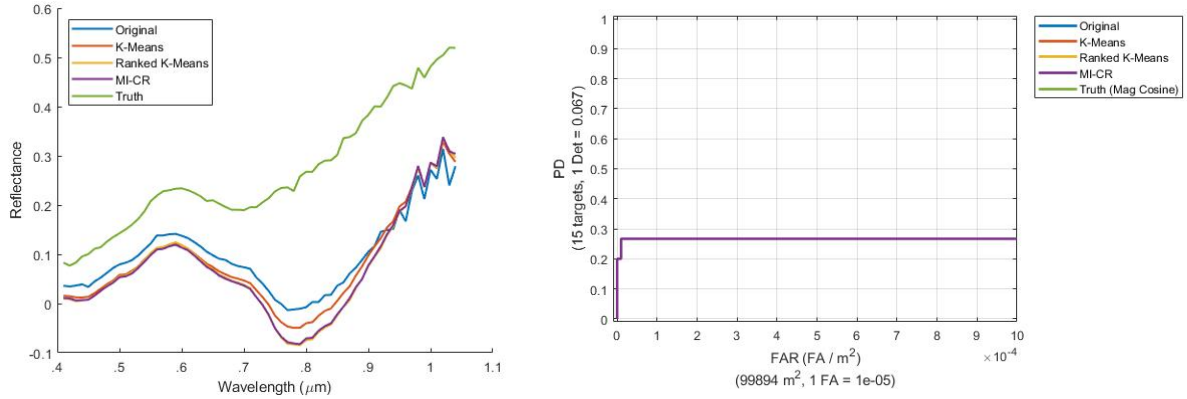
Method	(ACE) Time	(ACE) AUC	(SMF) Time	(SMF) AUC	(HSD) NAUC
Original	0.159	0.384	0.180	0.347	-
K-Means	0.011	0.386	0.013	0.351	-
Ranked K-Means	0.052	0.383	0.024	0.351	-
MI-CR	0.083	0.385	0.067	0.351	-
MI-HE	-	0.379	-	-	0.453
Truth	0	0.040	0	0.045	-

4.2.1.4 Pea green targets



(a) (b)

Figure 4-18. Experiment results for Multi-Target MI-ACE using different initialization techniques, on MUUFL Gulfport data, pea green targets. (a) Initialized pea green target signatures. (b) Pea green targets ROC curve.



(a) (b)

Figure 4-19. Experiment results for Multi-Target MI-SMF using different initialization techniques, on MUUFL Gulfport data, pea green targets. (a) Initialized pea green target signatures. (b) Pea green targets ROC curve.

Table 4-7. Pea green target initialization experiments showing the proposed methods for ACE, SMF, and the ground truth. The run times and NAUC performances on the MUUFL Gulfport hyperspectral dataset are listed. MI-HE results obtained from (Jiao, 2017).

Method	(ACE) Time	(ACE) AUC	(SMF) Time	(SMF) AUC	(HSD) NAUC
Original	0.191	0.267	0.185	0.267	-
K-Means	0.013	0.267	0.011	0.267	-
Ranked K-Means	0.174	0	0.030	0.267	-
MI-CR	0.173	0.267	0.100	0.267	-
MI-HE	-	0.267	-	-	0.267
Truth	0	0.267	0	0	-

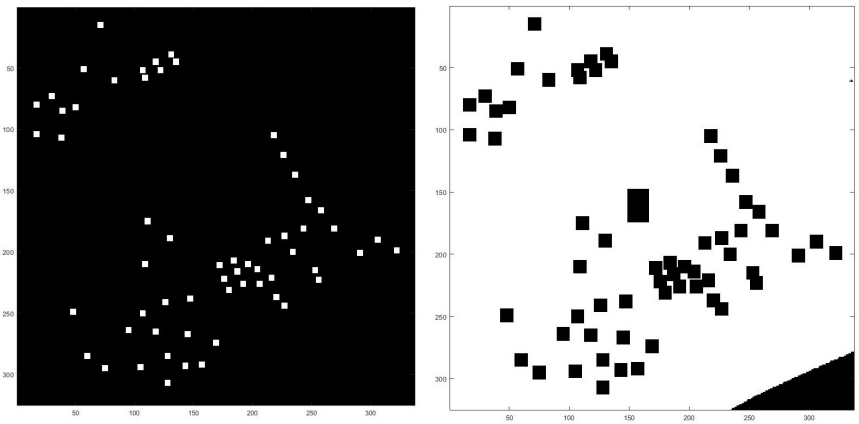
4.2.2 All Target Types Detection Experiments

For the experiments that follow, all of the target types are treated as the target class and the positive bags contain all of the target types. The algorithms are unaware which positive bags contain which target types, only that the positive bags contain a target. The "Truth" target signatures for each target type were manually extracted from the image to provide a performance comparison of the ground truth target signature versus the learned target signatures. The cosine similarity (normalized inner product) between the "Truth" signature and a test pixel is used to generate a confidence for every test pixel. When computing the "Truth" signature confidence for the SMF experiments, the cosine similarity is not normalized by the test pixel, similar to SMF. The proposed initialization and optimization methods are tested. The ACE or SMF detection statistic is used with the corresponding proposed ACE or SMF algorithms to determine the confidence of a test sample being a target.

The resulting confidences are scored using the bullwinkle scoring method outlined by Glenn et al. (2013) with a halo size of 2, which accounts for the inaccuracies of the recording GPS system in the test data. The bullwinkle scoring method takes the maximum value within a region around the ground truth, known as a halo, and uses the maximum value from the halo to score the results. Due to the background mean subtraction done during whitening, the proposed algorithms' learned target concepts are not like the true target concepts. The learned target signatures are dewhitened, but the mean subtraction is not simple to add back in, so the

concepts are compared with the mean background subtracted. With this, it is difficult to verify that the algorithms learn the "right" signatures by comparing them directly to the ground truth signatures. However, they can be verified by considering their general shape and how the background mean is expected to affect the resulting learned signatures. The ROC curve performance, as well as comparing the target signatures, will justify that the algorithms are learning the "right" signatures. The Normalized Area Under the Curve (NAUC) is computed within the window of 0 to 0.001 FAR/m^2 of the ROC.

Like the individual target type bagging process, to create each positive bag for training, a pixel window of size 5 x 5 around the corresponding GPS ground truth location was used for each target of interest. This window size was chosen to account for the GPS inaccuracy used for recording the ground truth. Then most of the remaining area is used as one large negative bag. Here, an additional two pixel wide keep out region is added around each of the positive bags, as well as a mask is added to remove the bottom right corner which was not originally valid data in the image. Lastly, the four large atmospheric correction targets in the middle of the image are masked out. The remaining region in white corresponds to the large negative bag used for training. The bagging process for all of the targets is showcased in Figure 4-20. Following, the number of targets and size is shown in Table 4-8.



(a) (b)

Figure 4-20. Bagging image masks for all targets in the MUUFL Gulfport dataset with a 5 x 5 window. The white region corresponds to the regions extracted during the bagging process. (a) Positive bag regions. (b) Negative bag region.

Table 4-8. The number of targets in the MUUFL Gulfport hyperspectral dataset for all targets.

Targets	Size: $0.5m$ x $0.5m$	Size: $1m$ x $1m$	Size: $3m$ x $3m$	All sizes
All targets	19	19	19	57

4.2.2.1 Initialization methods experiments

The optimal initialization hyperparameter settings were tested in Appendix B and are used in the following experiments. The experiments compare the five multi-target initialization methods proposed which include the original greedy method, the K-Means method, the Ranked K-Means method, the MI-CR method, and the uniqueness term method. Additionally, the single target version as well as using the extracted ground truth signal with a cosine similarity metric explained in Section 4.2.2 are compared. The multi-target initialization methods are selected to initialize four signatures. Both the ACE and SMF versions were tested. The target concepts for the multi-target MI-ACE and multi-target MI-SMF were used for detection with the ACE and SMF similarity statistic, respectively. The maximum ACE or SMF value, for all of the learned target concepts, was taken to be the confidence of detecting an unknown sample.

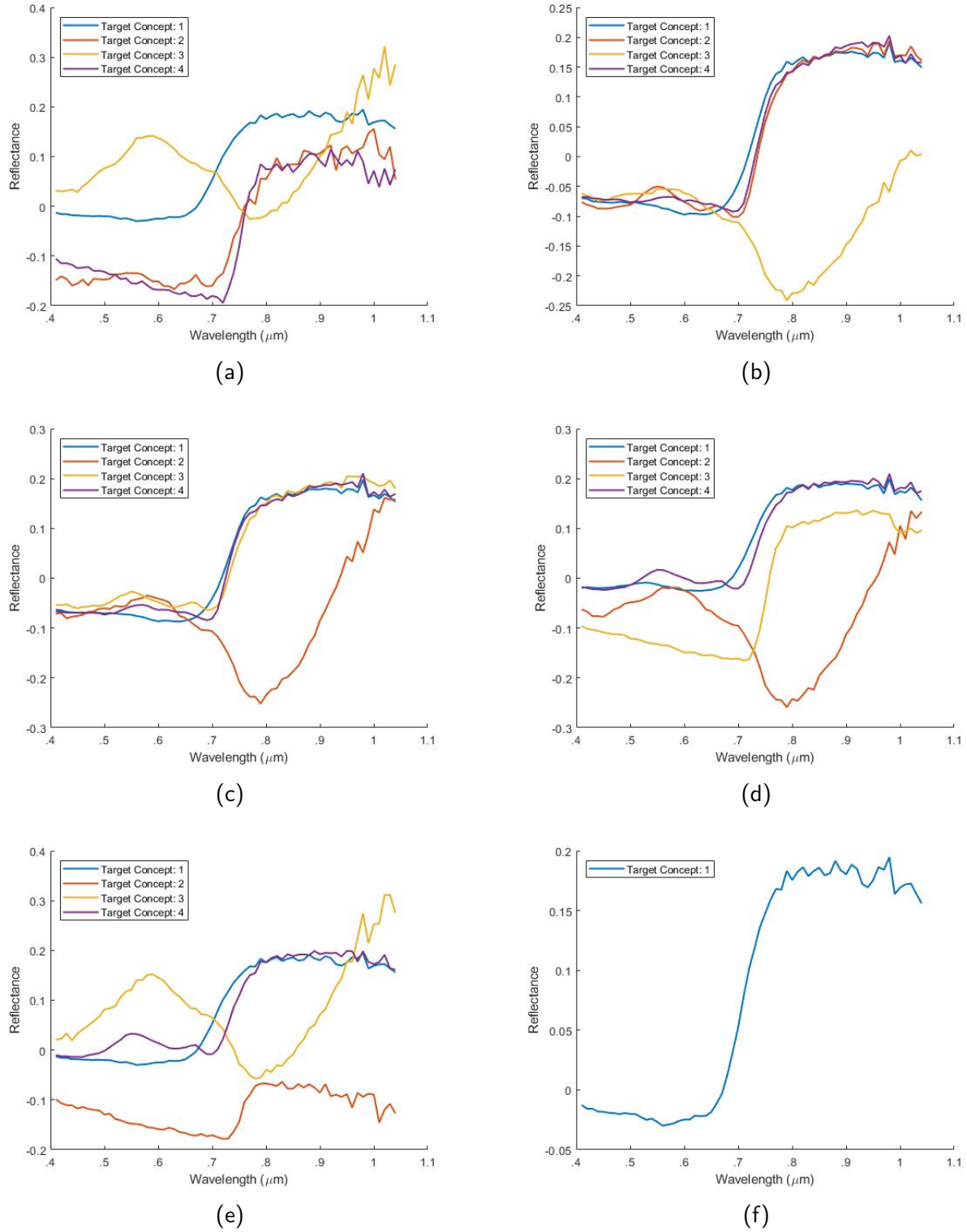


Figure 4-21. Initialized signatures for Multi-Target MI-ACE on MUUFL Gulfport data, all targets. (a) Original method. (b) K-Means method. (c) Ranked K-Means method. (d) MI-CR method. (e) Uniqueness term method. (f) Single target MI-ACE.

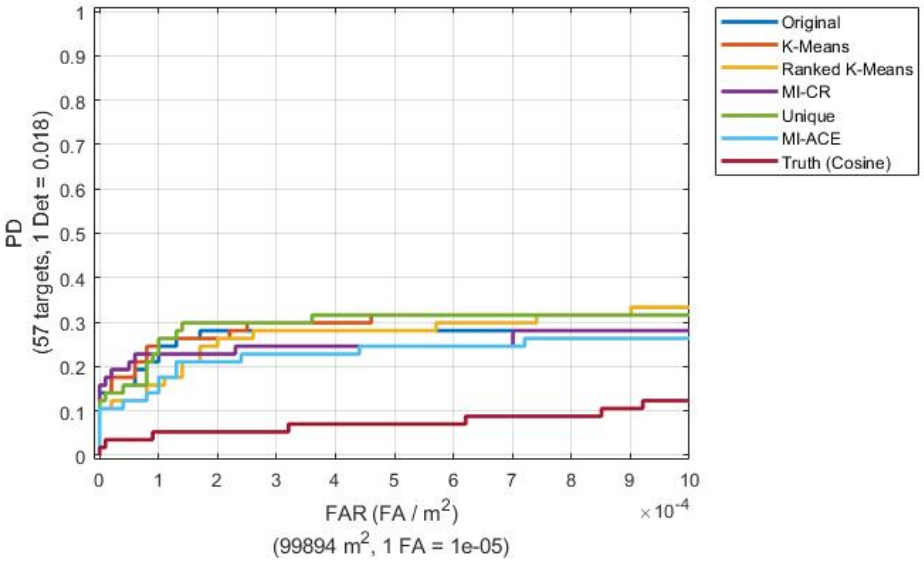


Figure 4-22. ROC curves for initialized signatures for different proposed initialization methods using (ACE) statistic with optimal parameters.

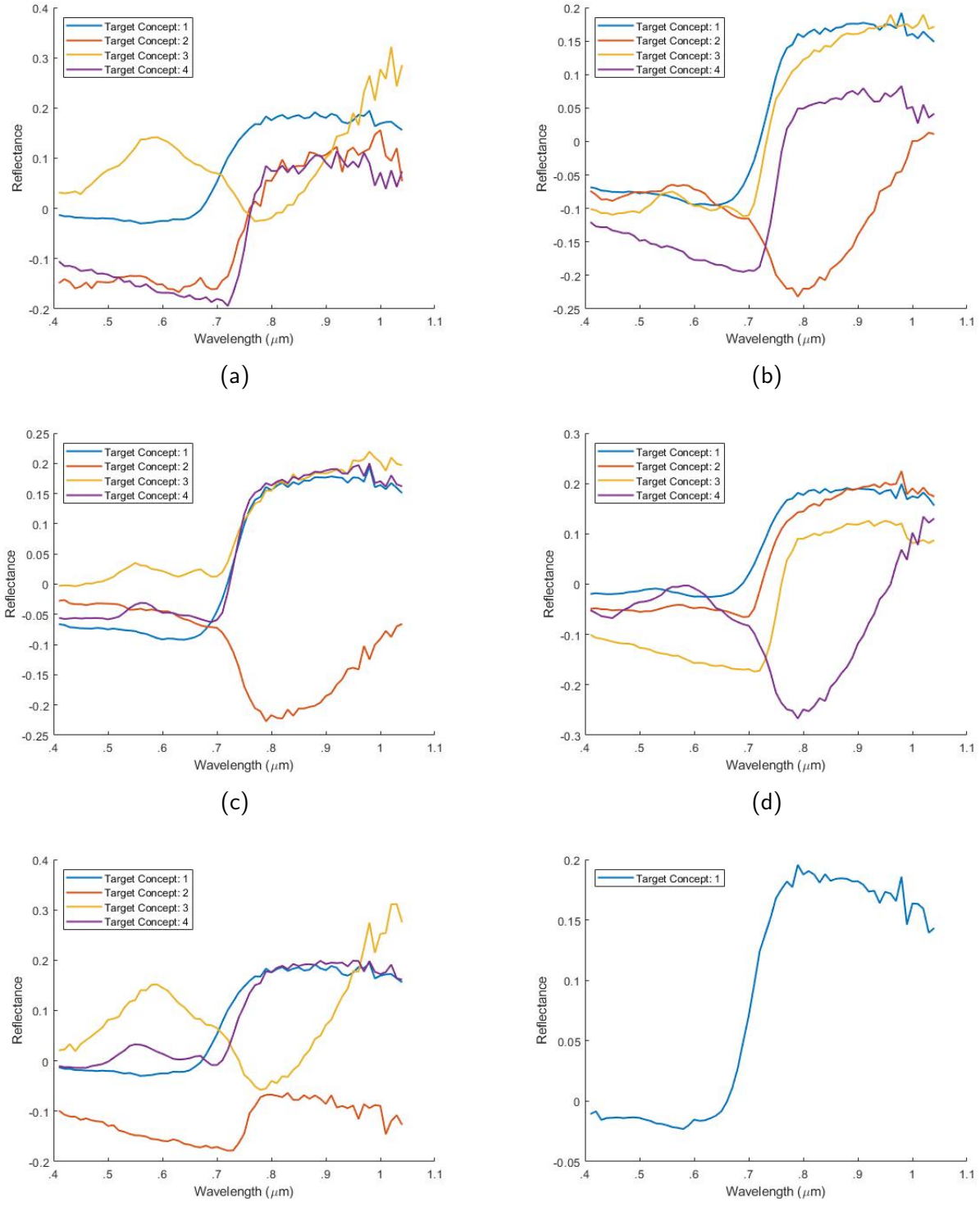


Figure 4-23. Initialized signatures for Multi-Target MI-SMF on MUUFL Gulfport data, all targets. (a) Original method. (b) K-Means method. (c) Ranked K-Means method. (d) MI-CR method. (e) Uniqueness term method. (f) Single target MI-SMF.

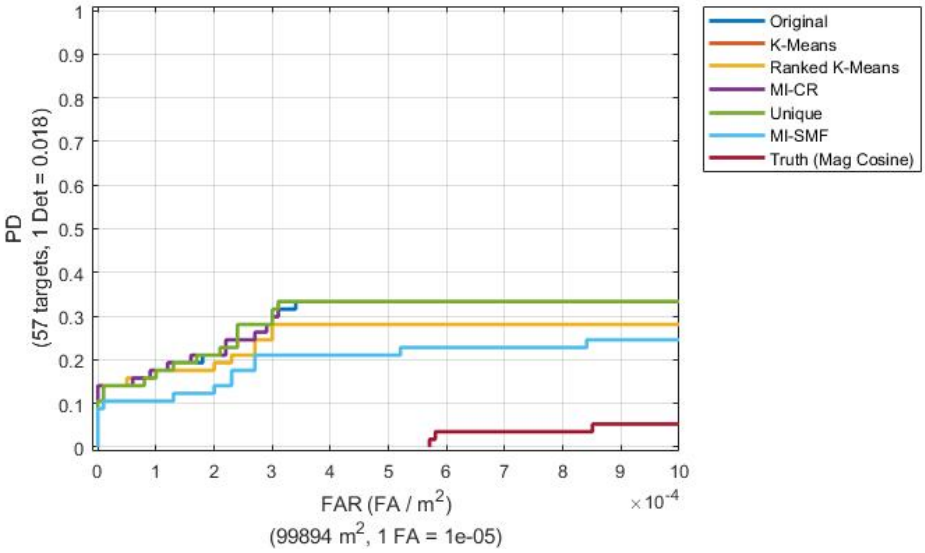


Figure 4-24. ROC curves for initialized signatures for different proposed initialization methods using (SMF) statistic with optimal parameters

Table 4-9. Comparison of the proposed initialization methods' run time and ROC NAUC using all of the target types for training and testing using the MUUFL Gulfport hyperspectral dataset. MI-HE results obtained from (Jiao, 2017).

Method	(ACE) Time	(ACE) AUC	(SMF) Time	(SMF) AUC	(HSD) NAUC
Original	7.09	0.268	7.08	0.292	-
K-Means	0.041	0.292	0.046	0.250	-
Ranked K-Means	0.060	0.271	0.074	0.293	-
MI-CR	1.22	0.250	1.374	0.292	-
Unique	7.09	0.295	7.12	0.292	-
MI-HE	-	0.257	-	-	0.304
MI-ACE/SMF	1.71	0.228	1.75	0.198	-
Truth	0	0.074	0	0.018	-

4.2.2.2 Optimization methods experiments

The optimal optimization hyperparameter settings were tested in Appendix B and are used in the following experiments. The experiments that follow compare the four multi-target optimization methods proposed which include the original optimization method, the weighted optimization method, the weighted kernel method, and the uniqueness term method. Additionally, the single target version of MI-ACE and MI-SMF, as well as using the extracted ground truth signal with a cosine similarity metric explained in Section 4.2.2, are compared with the proposed optimization algorithms. The multi-target methods are selected to initialize four signatures using the original greedy method and then optimize those signatures with the varying optimization methods. Both the ACE and SMF versions were tested. Figures **??** - **??** show the optimized signatures for all of the methods with their respective ROC curves for both ACE and SMF.

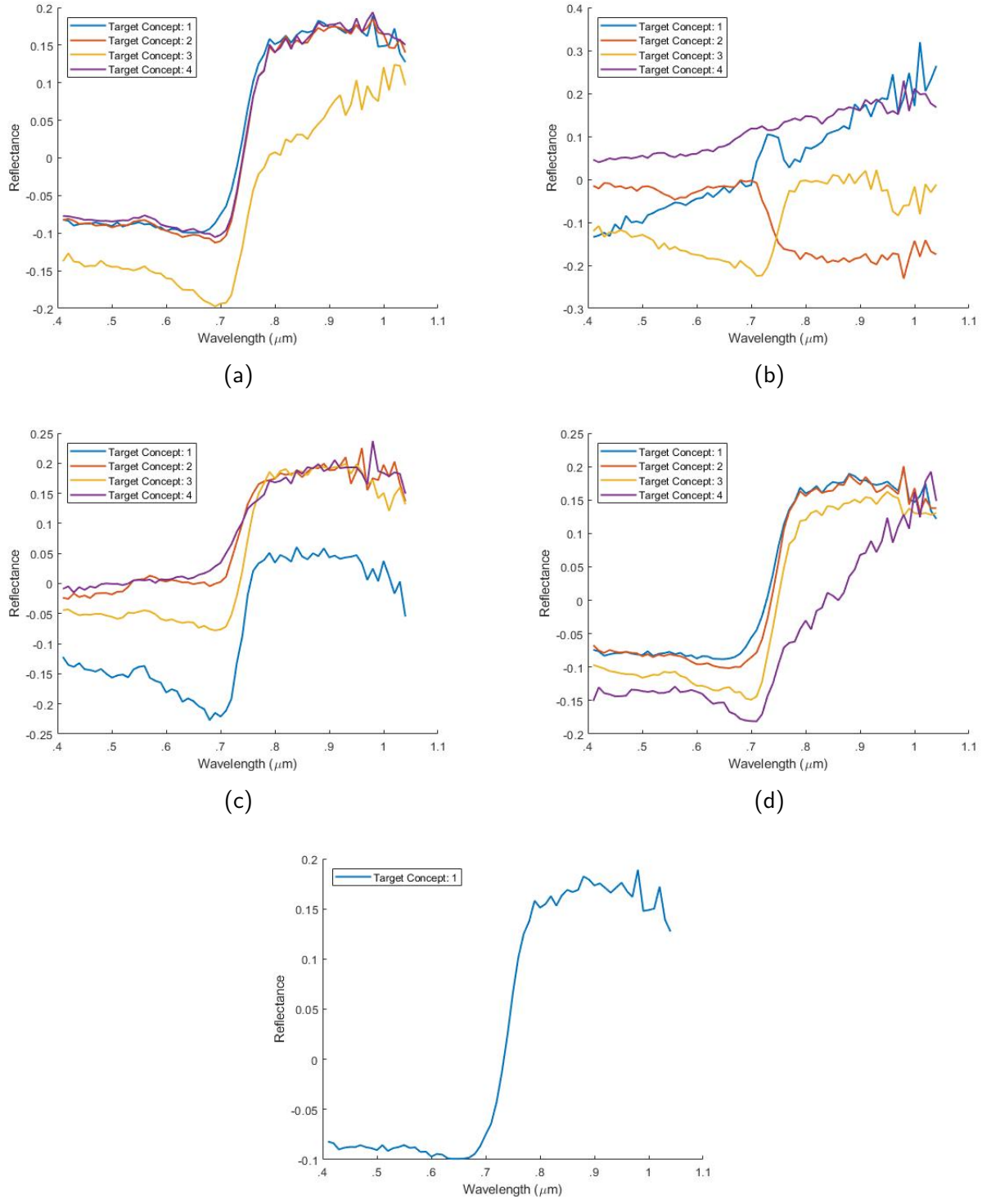


Figure 4-25. Optimized signatures for Multi-Target MI-ACE on MUUFL Gulfport data, all targets. (a) Original method. (b) Weighted method. (c) Weighted kernel method. (d) Uniqueness term method. (e) Single target MI-ACE.

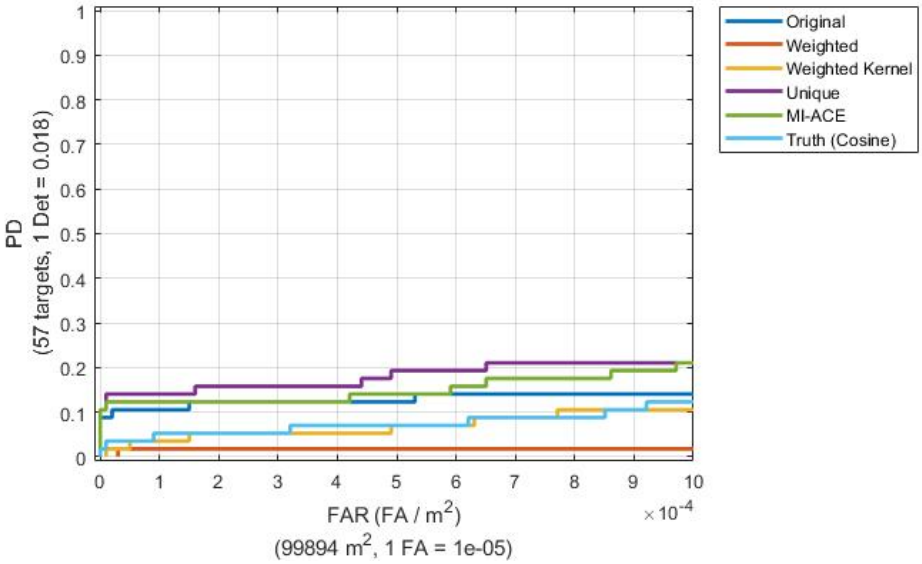


Figure 4-26. ROC curves for optimized signatures for different proposed optimization methods using (ACE) statistic with optimal parameters.

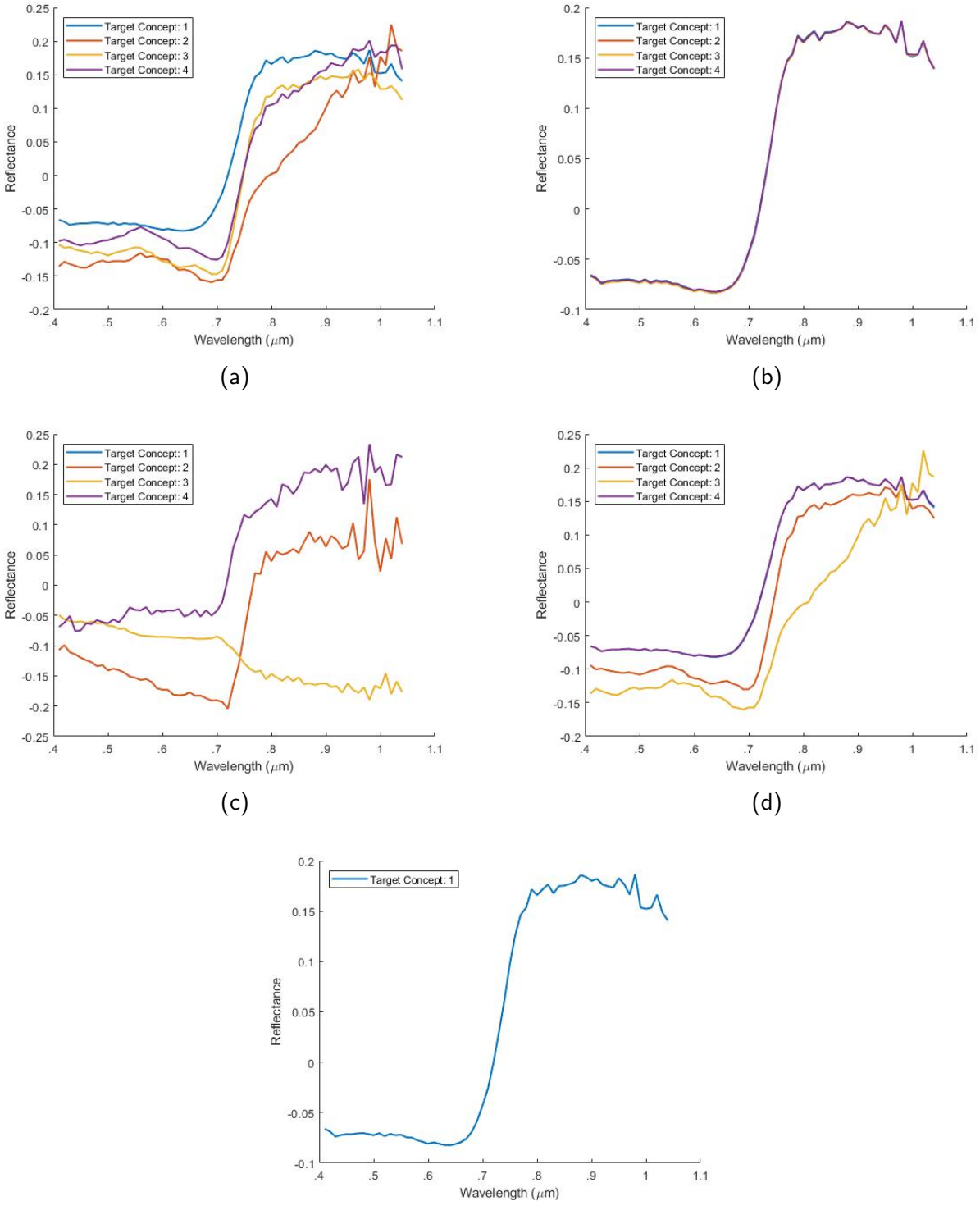


Figure 4-27. Optimized signatures for Multi-Target MI-SMF on MUUFL Gulfport data, all targets. (a) Original method. (b) Weighted method. (c) Weighted kernel method. (d) Uniqueness term method. (e) Single target MI-SMF.

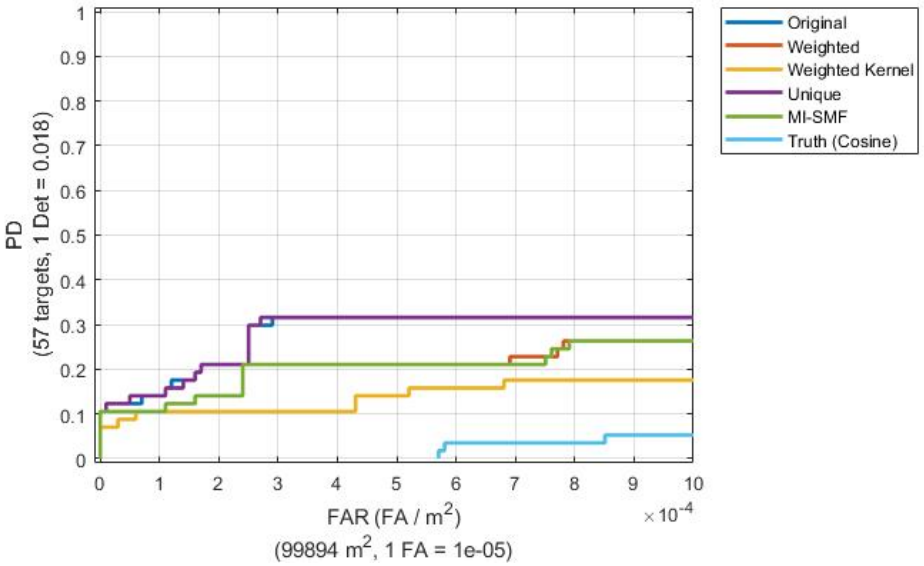


Figure 4-28. ROC curves for optimized signatures for different proposed optimization methods using (SMF) statistic with optimal parameters.

Table 4-10. Comparison of the proposed optimization methods' ROC NAUC using all of the target types for training and testing using the MUUFL Gulfport hyperspectral dataset. MI-HE results obtained from (Jiao, 2017).

Method	(ACE) NAUC	(SMF) NAUC	(HSD) NAUC
Original	0.128	0.278	-
Weighted	0.017	0.202	-
Weighted kernel	0.068	0.138	-
Unique	0.180	0.277	-
MI-HE	0.257	-	0.304
MI-ACE/SMF	0.149	0.201	-
Truth	0.074	0.018	-

CHAPTER 5
CONCLUSIONS AND FUTURE WORK

In this work, various methods were proposed and investigated for the multi-target MI-ACE and multi-target MI-SMF algorithms. Through the experiments, it was seen that these algorithms were able to learn multiple target concepts and improve performance when there were multiple target types in training and test sets.

The experiments in Chapter 4 showcase the different functionalities of the various proposed multi-target algorithms. It was seen that in many cases the proposed algorithms would maintain or exceed performance of the single target version. Additionally, some of the proposed initialization clustering methods significantly reduce the run time of initializing signatures, namely, the K-Means, ranked K-Means, and MI-CR algorithms. Of these clustering techniques it was seen that the K-Means technique provided the most consistent results, and would often outperform the greedy method using the original objective function. It was also seen through the experiments in Chapter 4 and Appendix A and B, that the uniqueness term method performed as expected. With this method, the learned signatures were encouraged to be different through the objective function, and were seen to be so in the results shown.

It was also seen through the experiments in Appendix A and B, that the performance was not especially sensitive to the number of clusters, so long as there were enough clusters to classify the diverse background of the MUUFL Gulfport dataset. If a background is to be expected to be uniform, a lower number of clusters may be used and the same performance can be expected.

Lastly, it was seen that the weighted and kernel weighted optimization methods did not perform well. After optimization, the signatures did not look like the extracted ground truth signatures. It is believed that the weights are updating the target concepts to be an average of many different target types. Rather, it is desired for the weights to only update the target concepts towards a specific target type.

In conclusion, the greedy method which uses the original objective function, the uniqueness term method, as well as the K-Means initialization method proved to be the most consistent in performance and robust to hyperparameter changes. The other methods provided better performance for some of the test cases, but were overall not consistent. The research question to determine a way to optimize a group of target concepts within this framework is still open to investigation. This problem is especially challenging. It is desired to be able to optimize the signatures to be unique and accurately representative of the various target types. The original optimization method showcased the best optimization performance, but it is believed this can be improved if an optimization technique can learn the target type labels while performing optimization.

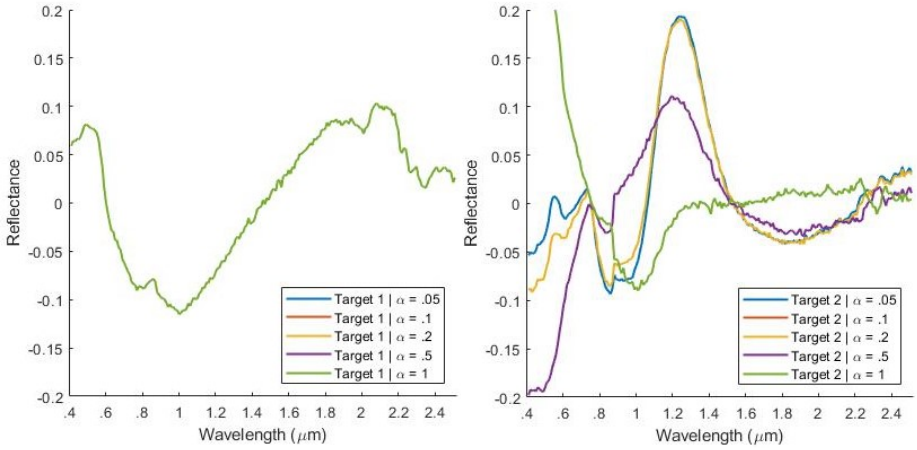
In the MUUFL Gulfport all targets optimization experiment, the optimization caused the performance of the proposed algorithms to decrease. This demonstrates the need to develop other optimization approaches that can optimize each target individually to at least maintain performance of the initialized signatures. Potential future work includes investigating an Expectation Maximization optimization that directly learns the latent target type variables of the true positive instances. With this, the target concepts could be updated using the original single target update equation shown in Equation (2-28), by only considering the positive instances that are expected to be in the same target type class. Additionally, an optimization method to update the target signatures using only the positive bag representatives that are the most similar to the target concept being updated is expected to improve optimization performance.

One technique that will be investigated is including a maximum operation during optimization, where only the positive bag representatives, $\mathbf{x}_{j,k}$, that are the most similar to each target being optimized is used during that target's optimization. For example, if positive bag j is chosen by target k during optimization, because it's positive bag representative, $\mathbf{x}_{j,k}$, is the most similar to target k, then no other other target concepts being optimized will be able to use a positive bag representative from that bag during optimization. This will address the

fact that not all positive bags have every target concept in them. With this, it is expected that optimization performance will increase significantly by allowing each target concept to optimize with only the positive bag representatives that are the same target type.

APPENDIX A
SIMULATED DATA HYPERPARAMETER EXPERIMENTS

A.1 Initialization Hyperparameter Experiments



(a) (b)

Figure A-1. Two initialized signatures for Multi-Target MI-ACE using unique targets with various α levels on simulated data. (a) Estimated target signature 1. (b) Estimated target signature 2.

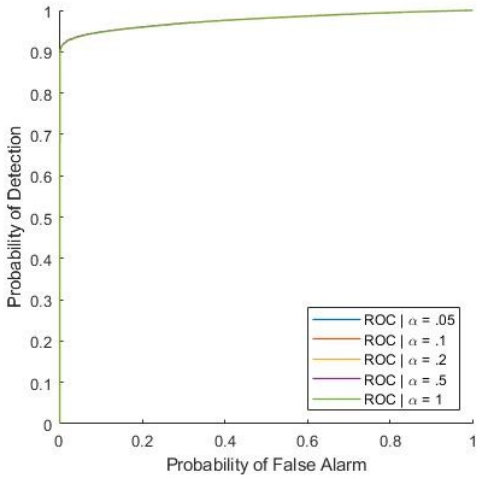
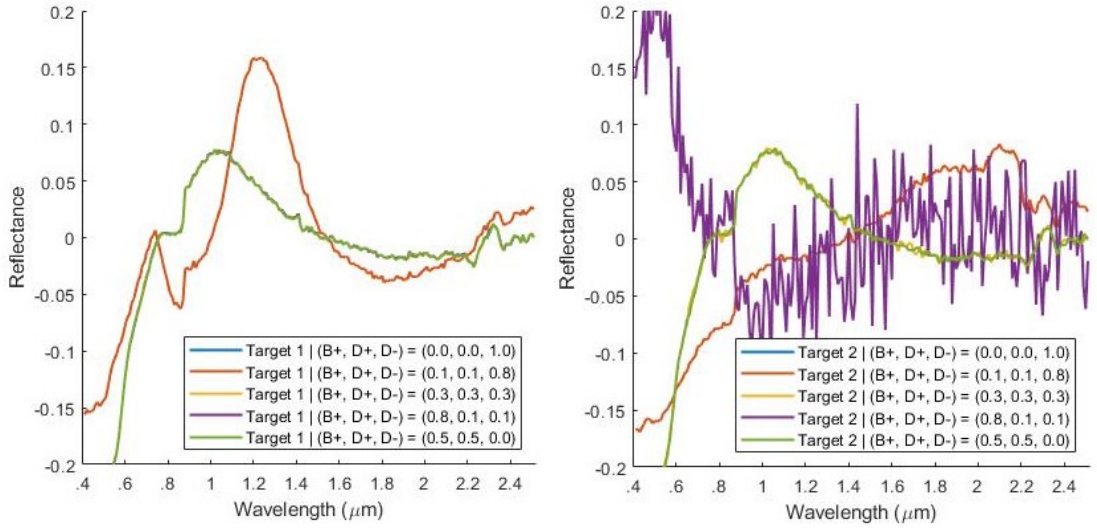


Figure A-2. ROC curves for two initialized signatures using Multi-Target MI-ACE with unique target initialization with various α levels on simulated data

Table A-1. Uniqueness term initialization hyperparameter experiment results showing the run time and ROC AUC.

Uniqueness term	Run time	ROC AUC
$\alpha = .05$	6.36	0.976
$\alpha = .1$	6.20	0.976
$\alpha = .2$	6.17	0.976
$\alpha = .5$	6.18	0.977
$\alpha = 1$	6.20	0.976



(a) (b)

Figure A-3. Two initialized signatures for Multi-Target MI-ACE using ranked K-Means with various cluster rank weights, (wB^+, wD^+, wD^-), on simulated data with 4 targets per positive bag. (a) Estimated target signature 1. (b) Estimated target signature 2.

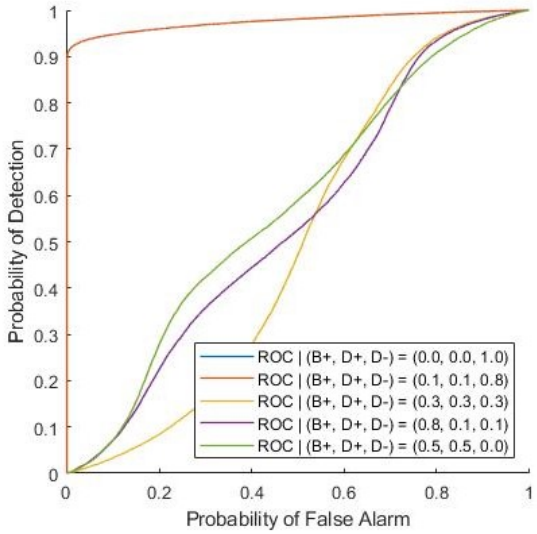
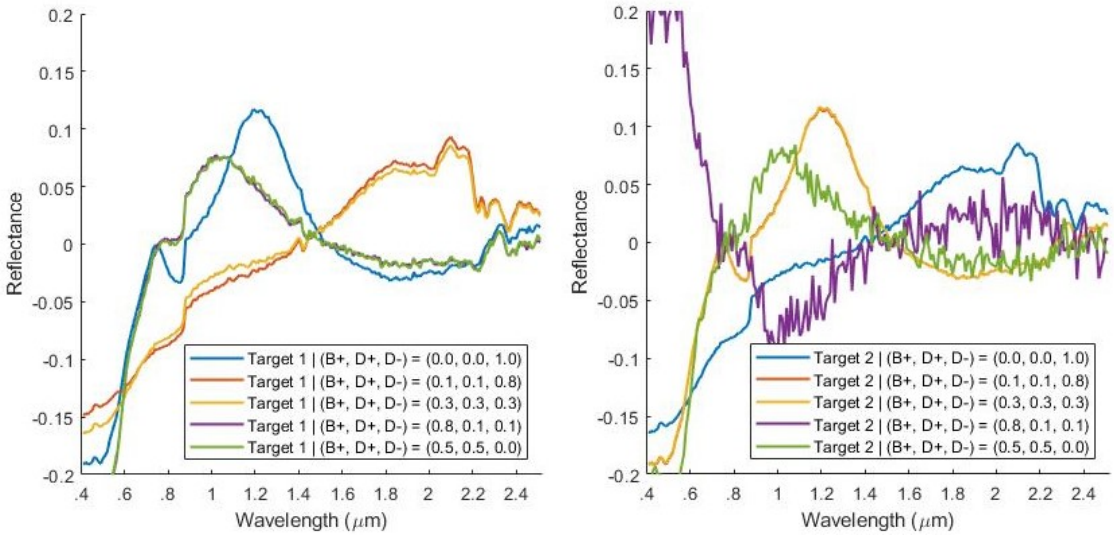


Figure A-4. ROC curves for two initialized signatures for Multi-Target MI-ACE using ranked K-Means with various cluster rank weights, (wB^+, wD^+, wD^-), on simulated data with 4 targets per positive bag



(a) (b)

Figure A-5. Two initialized signatures for Multi-Target MI-ACE using ranked K-Means with various cluster rank weights, (wB^+, wD^+, wD^-), on simulated data with 15 targets per positive bag. (a) Estimated target signature 1. (b) Estimated target signature 2.

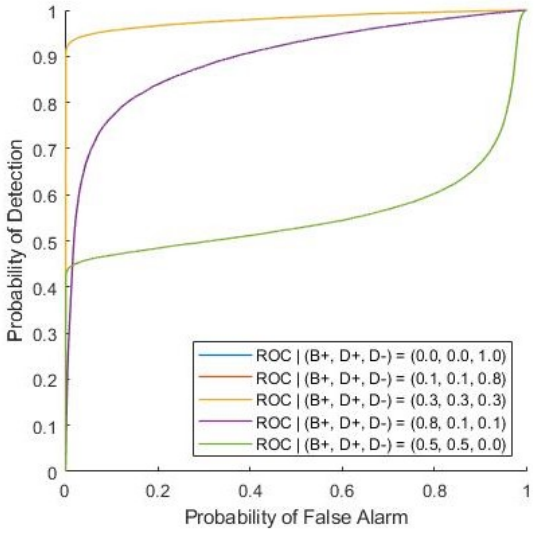
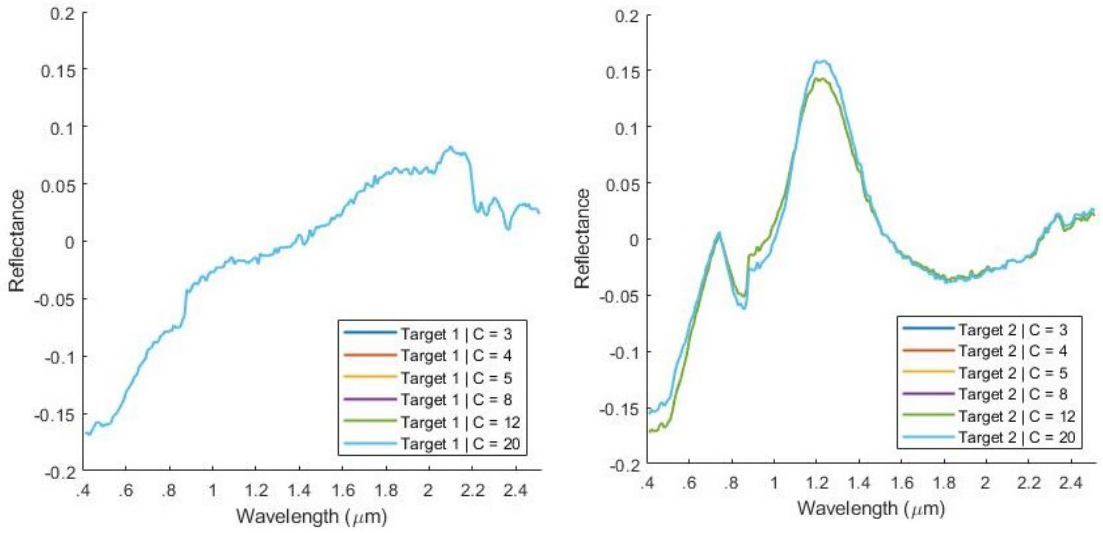


Figure A-6. ROC curves for two initialized signatures for Multi-Target MI-ACE using ranked K-Means with various cluster rank weights, (wB^+, wD^+, wD^-), on simulated data with 15 targets per positive bag

Table A-2. Cluster rank weights initialization hyperparameter experiment results showing the run time and ROC AUC for two different training sets, one with 4 target per positive bag, and the later with 15 targets per positive bag.

	4 Targets per + bag		15 Targets per + bag	
Weights: (wB+, wD+, wD-)	Run time	ROC AUC	Run time	ROC AUC
(0, 0, 1)	0.327	0.976	0.299	0.980
(.1, .1, .8)	0.181	0.976	0.127	0.980
(.33, .33, .33)	0.116	0.497	0.111	0.980
(.8, .1, .1)	0.097	0.545	0.080	0.895
(.5, .5, 0)	0.121	0.574	0.111	0.555



(a) (b)

Figure A-7. Two initialized signatures for Multi-Target MI-ACE using original multi-target objective function with various number of clusters, C, on simulated data. (a) Estimated target signature 1. (b) Estimated target signature 2.

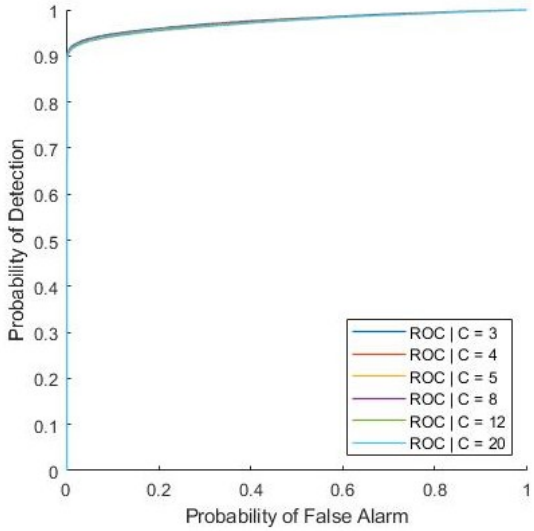
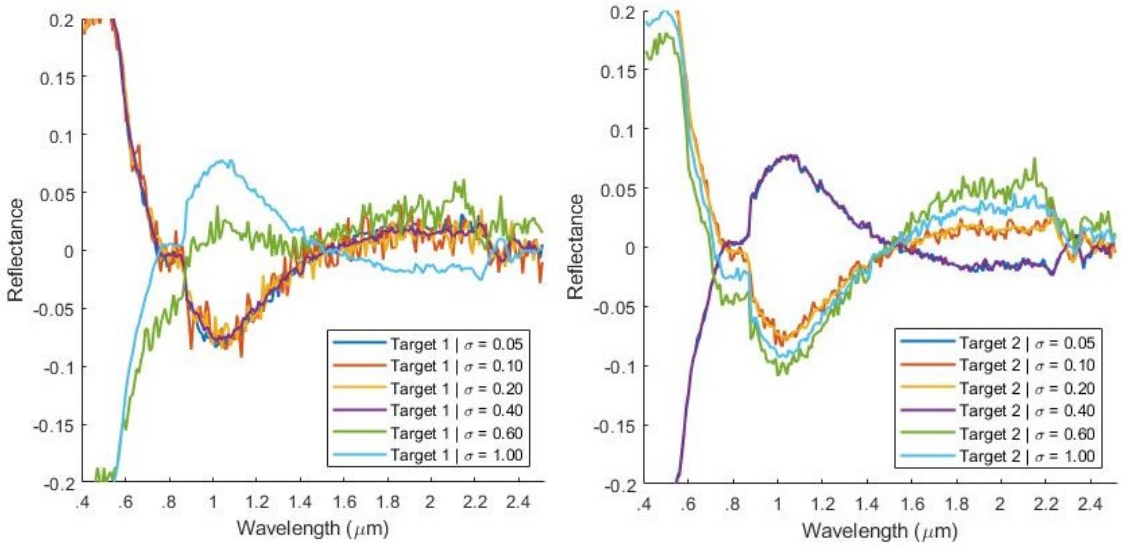


Figure A-8. ROC curves for two initialized signatures for Multi-Target MI-ACE using original multi-target objective function with various number of clusters, C, on simulated data

Table A-3. Number of clusters initialization hyperparameter experiment while using K-Means. Results show the run time and ROC AUC.

Number of clusters	Run time	ROC AUC
$C = 3$	0.269	0.976
$C = 4$	0.129	0.976
$C = 5$	0.095	0.976
$C = 8$	0.088	0.976
$C = 12$	0.120	0.976
$C = 20$	0.097	0.976

A.2 Optimization Hyperparameter Experiments



(a) (b)

Figure A-9. Two optimized signatures for Multi-Target MI-ACE using kernel based weighted optimization with various bandwidths, σ, on simulated data. (a) Estimated target signature 1. (b) Estimated target signature 2.

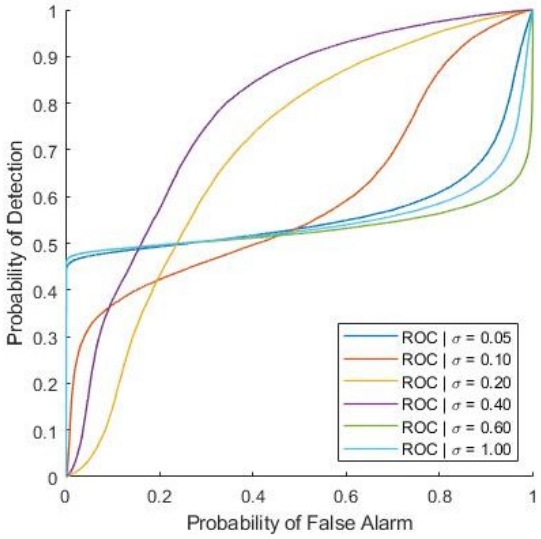
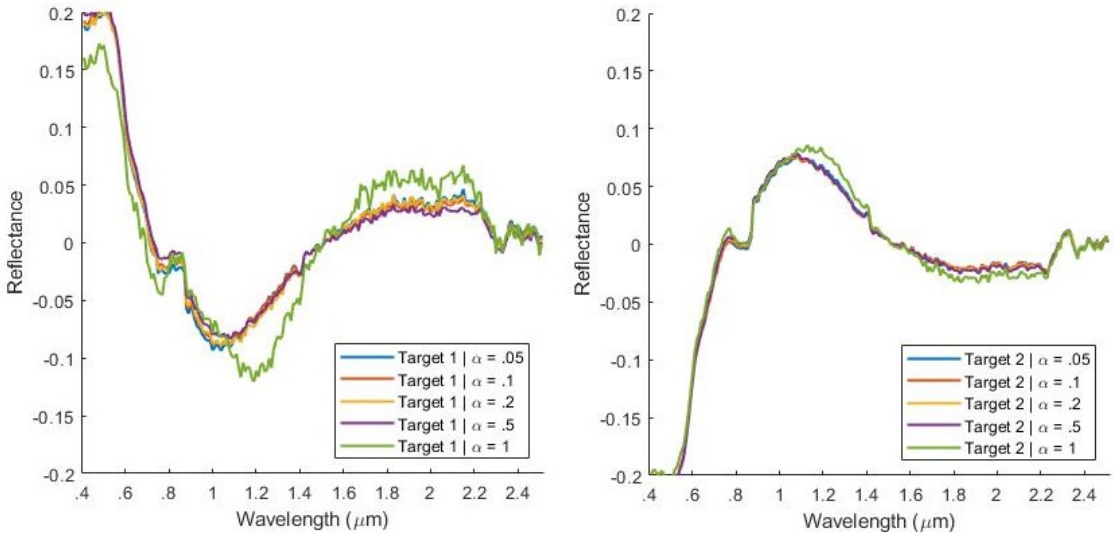


Figure A-10. ROC curves for two optimized signatures for Multi-Target MI-ACE using kernel based weighted optimization with various bandwidths, σ, on simulated data

Table A-4. Kernel bandwidth hyperparameter optimization experiment. Results show ROC AUC for the weighted kernel optimization method.

Bandwidth	ROC AUC
$\sigma = .05$	0.565
$\sigma = .1$	0.600
$\sigma = .2$	0.697
$\sigma = .4$	0.781
$\sigma = .6$	0.533
$\sigma = 1$	0.551



(a) (b)

Figure A-11. Optimized target signatures with various uniqueness term, α, values. Initialized signatures were from original multi-target initialization technique. (a) Estimated target signature 1. (b) Estimated target signature 2.

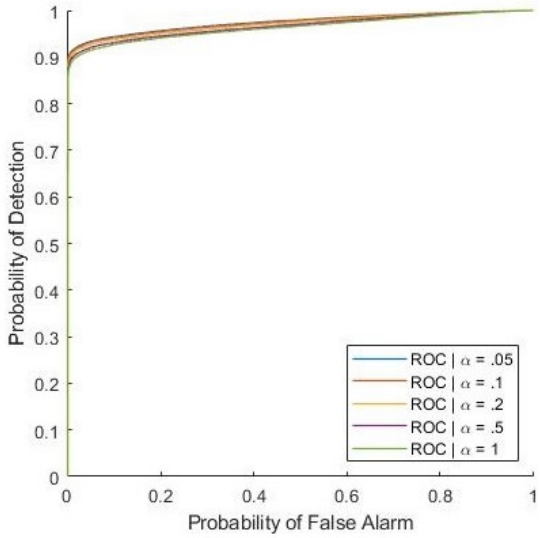


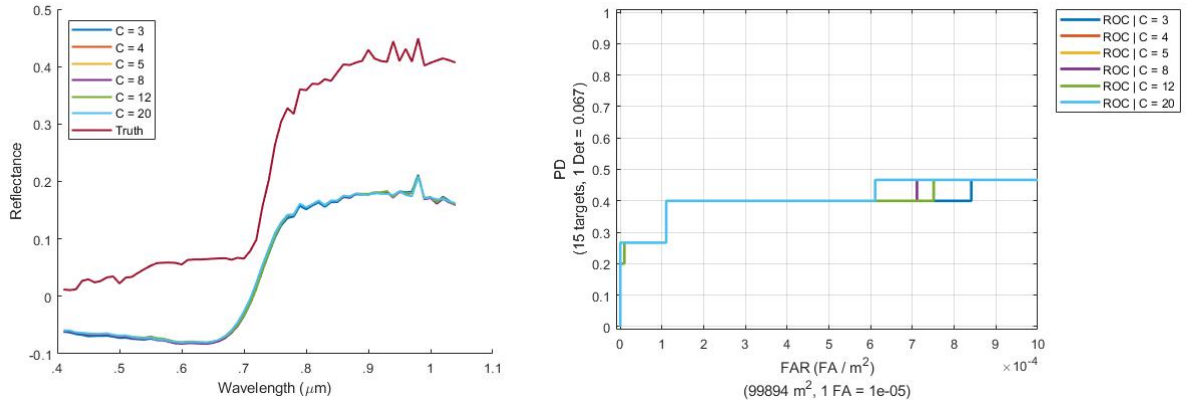
Figure A-12. ROC curves for optimized target signatures with various uniqueness term, α, values. Initialized signatures were from original multi-target initialization technique.

Table A-5. Uniqueness term hyperparameter experiment. Results show ROC AUC for the optimized signatures using the uniqueness term objective function.

Uniqueness term	ROC AUC
$\alpha = .05$	0.975
$\alpha = .1$	0.974
$\alpha = .2$	0.971
$\alpha = .5$	0.967
$\alpha = 1$	0.964

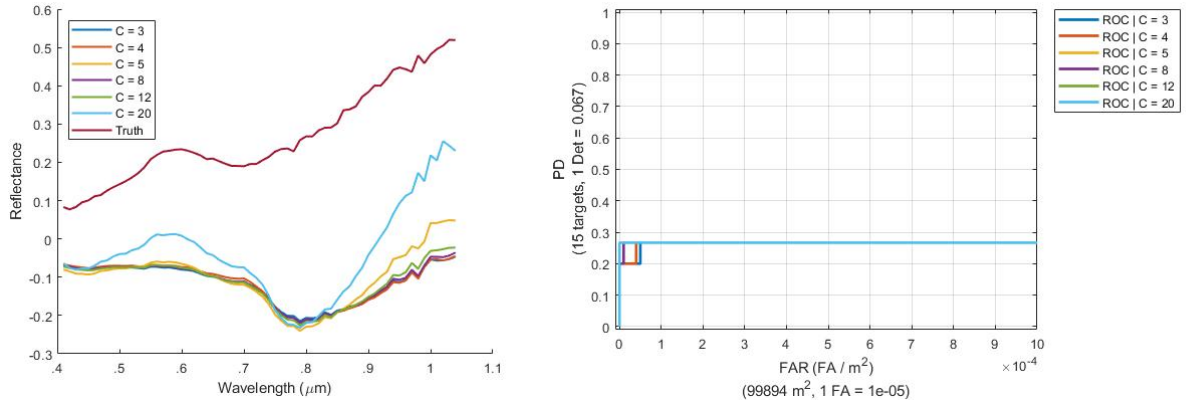
APPENDIX B
MUUFL GULFPORT HYPERSPECTRAL HYPERPARAMETER EXPERIMENTS

B.1 Single Target Initialization Hyperparameter Experiments



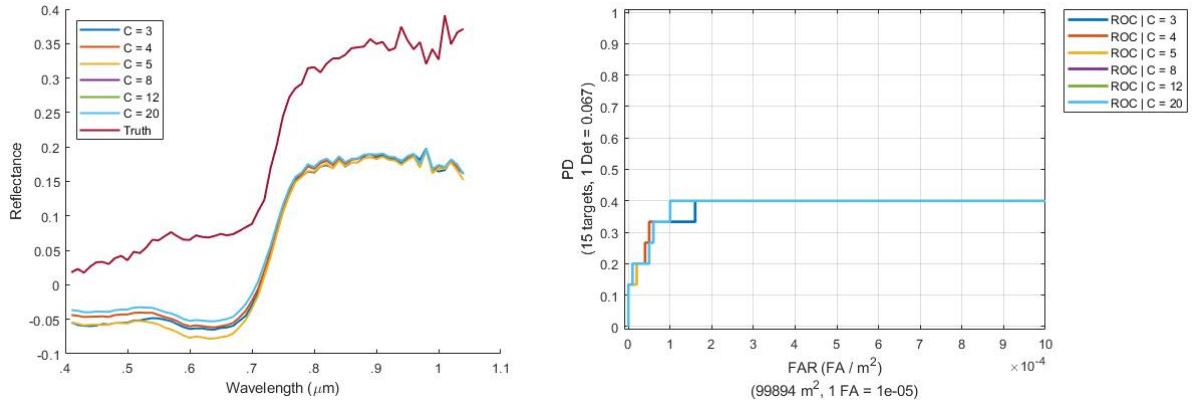
(a) (b)

Figure B-1. Experiment results for Multi-Target MI-ACE using K-Means with various number of clusters, C, on MUUFL Gulfport data, brown. (a) Estimated brown target signatures. (b) Brown targets ROC curve.



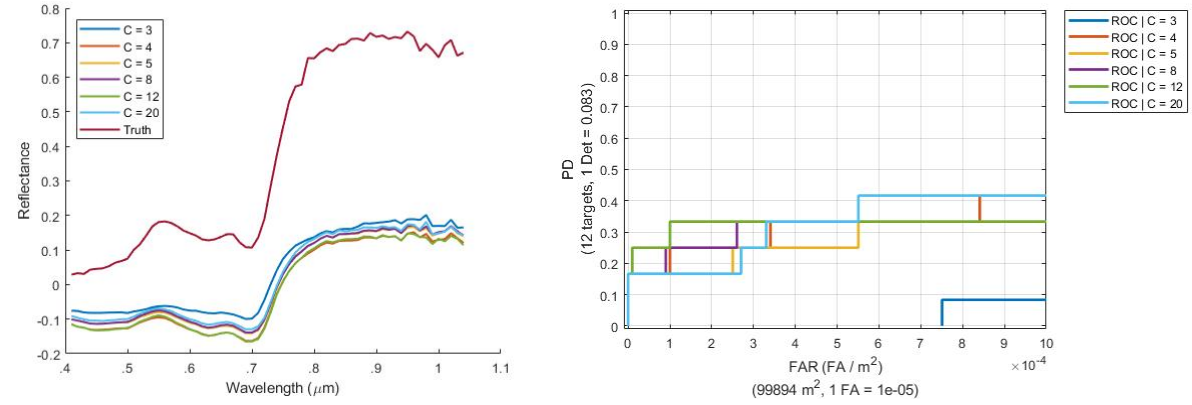
(a) (b)

Figure B-2. Experiment results for Multi-Target MI-ACE using K-Means with various number of clusters, C, on MUUFL Gulfport data, pea green. (a) Estimated pea green target signatures. (b) Pea green targets ROC curve.



(a) (b)

Figure B-3. Experiment results for Multi-Target MI-ACE using K-Means with various number of clusters, C, on MUUFL Gulfport data, dark green. (a) Estimated dark green target signatures. (b) Dark green targets ROC curve.

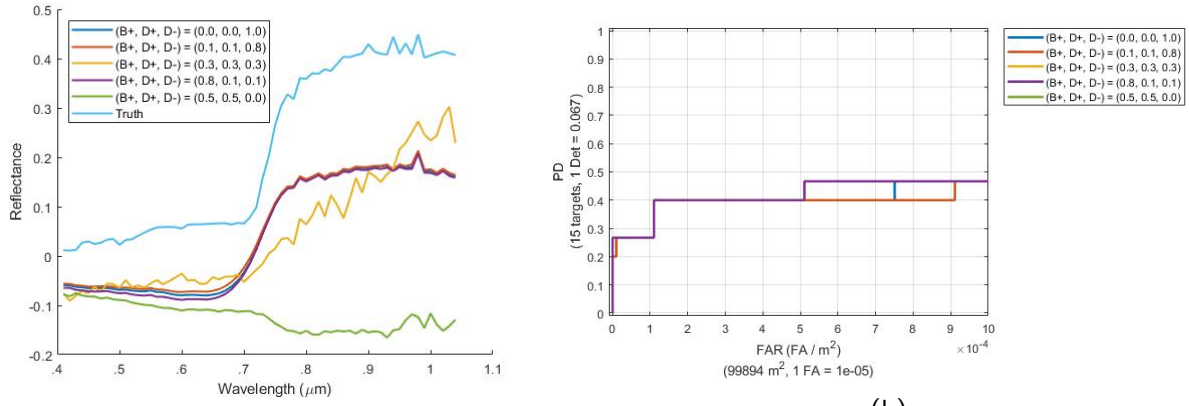


(a) (b)

Figure B-4. Experiment results for Multi-Target MI-ACE using K-Means with various number of clusters, C, on MUUFL Gulfport data, faux vineyard green. (a) Estimated faux vineyard green target signatures. (b) Faux vineyard green targets ROC curve.

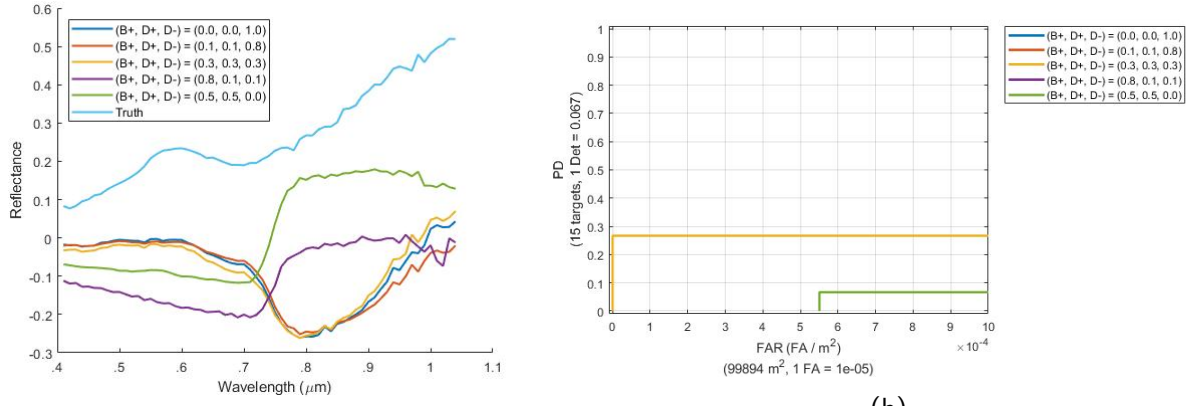
Table B-1. Number of clusters hyperparameter experiment using K-Means. Showing results with run time and AUC NAUC for learning a single target on each of the target types: brown, pea green, dark green, and faux vineyard green.

	Brown		Pea Green		Dark Green		Vineyard Green	
Number of clusters	Time	NAUC	Time	NAUC	Time	NAUC	Time	NAUC
$C = 3$	0.045	0.395	0.014	0.263	0.059	0.381	0.014	0.021
$C = 4$	0.016	0.401	0.014	0.264	0.012	0.387	0.012	0.310
$C = 5$	0.012	0.401	0.013	0.267	0.012	0.385	0.012	0.267
$C = 8$	0.015	0.405	0.015	0.266	0.014	0.385	0.014	0.304
$C = 12$	0.017	0.401	0.018	0.267	0.017	0.385	0.016	0.324
$C = 20$	0.022	0.411	0.022	0.267	0.022	0.385	0.021	0.321



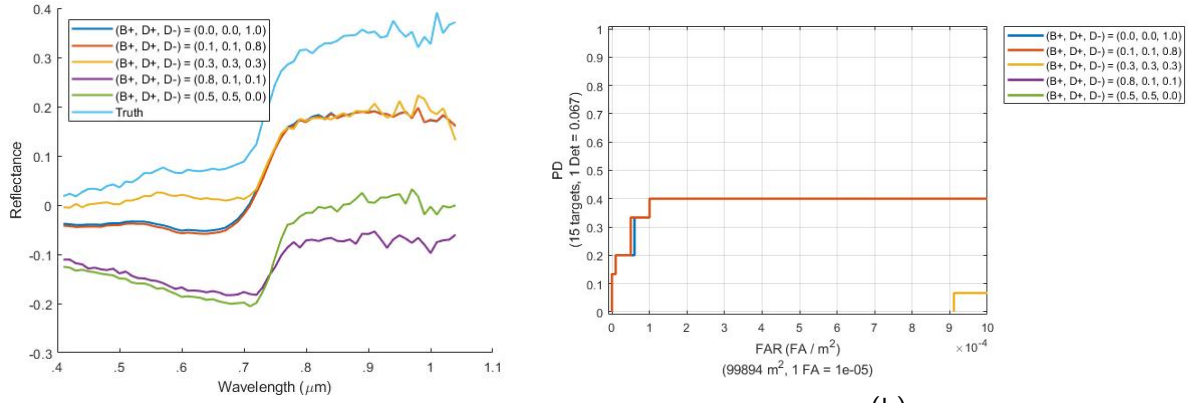
(a) (b)

Figure B-5. Initialized signatures and ROC Curve for Multi-Target MI-ACE using Ranked K-Means with various cluster rank weights, ($wB+$, $wD+$, $wD-$), on MUUFL Gulfport data, brown. (a) Estimated brown target signatures. (b) Brown targets ROC curve.



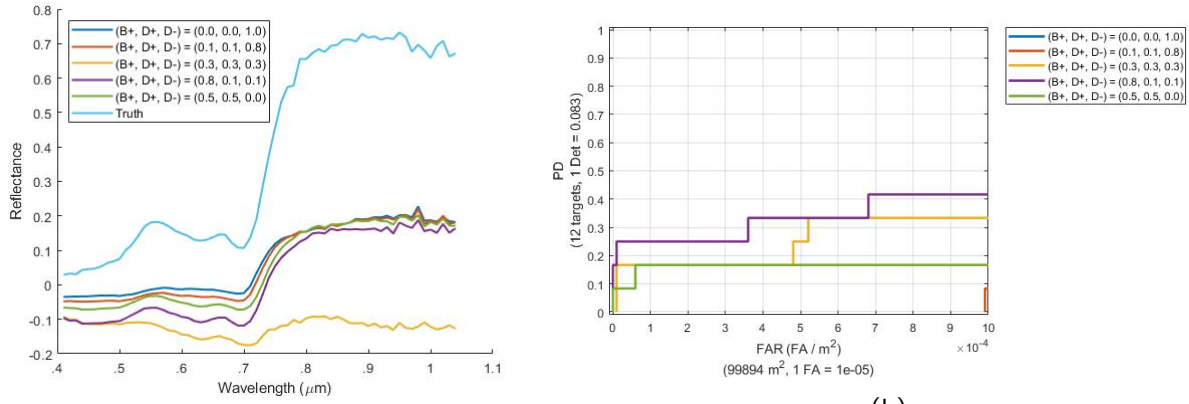
(a) (b)

Figure B-6. Experiment results for Multi-Target MI-ACE using Ranked K-Means with various cluster rank weights, ($wB+$, $wD+$, $wD-$), on MUUFL Gulfport data, pea green. (a) Estimated pea green target signatures. (b) Pea green targets ROC curve.



(a) (b)

Figure B-7. Experiment results for Multi-Target MI-ACE using Ranked K-Means with various cluster rank weights, ($wB+$, $wD+$, $wD-$), on MUUFL Gulfport data, dark green. (a) Estimated dark green target signatures. (b) Dark green targets ROC curve.



(a) (b)

Figure B-8. Experiment results for Multi-Target MI-ACE using Ranked K-Means with various cluster rank weights, ($wB+$, $wD+$, $wD-$), on MUUFL Gulfport data, faux vineyard green. (a) Estimated faux vineyard green target signatures. (b) Faux vineyard green targets ROC curve.

Table B-2. Cluster rank weights hyperparameter experiment. Showing results with run time and AUC NAUC for learning a single target on each of the target types: brown, pea green, dark green, and faux vineyard green.

	Brown		Pea Green		Dark Green		Vineyard Green	
(wB+, wD+, wD-)	Time	NAUC	Time	NAUC	Time	NAUC	Time	NAUC
(0, 0, 1)	0.077	0.401	0.041	0.267	0.049	0.385	0.043	0
(.1, .1, .8)	0.040	0.391	0.038	0.267	0.039	0.386	0.043	0
(.3, .3, .3)	0.034	0	0.039	0.267	0.046	0.006	0.037	0.248
(.8, .1, .1)	0.036	0.418	0.033	0	0.038	0	0.034	0.329
(.5, .5, 0)	0.050	0	0.037	0.030	0.033	0	0.043	0.162

B.2 All Targets Experiments

B.2.1 Initialization Hyperparameter Experiments

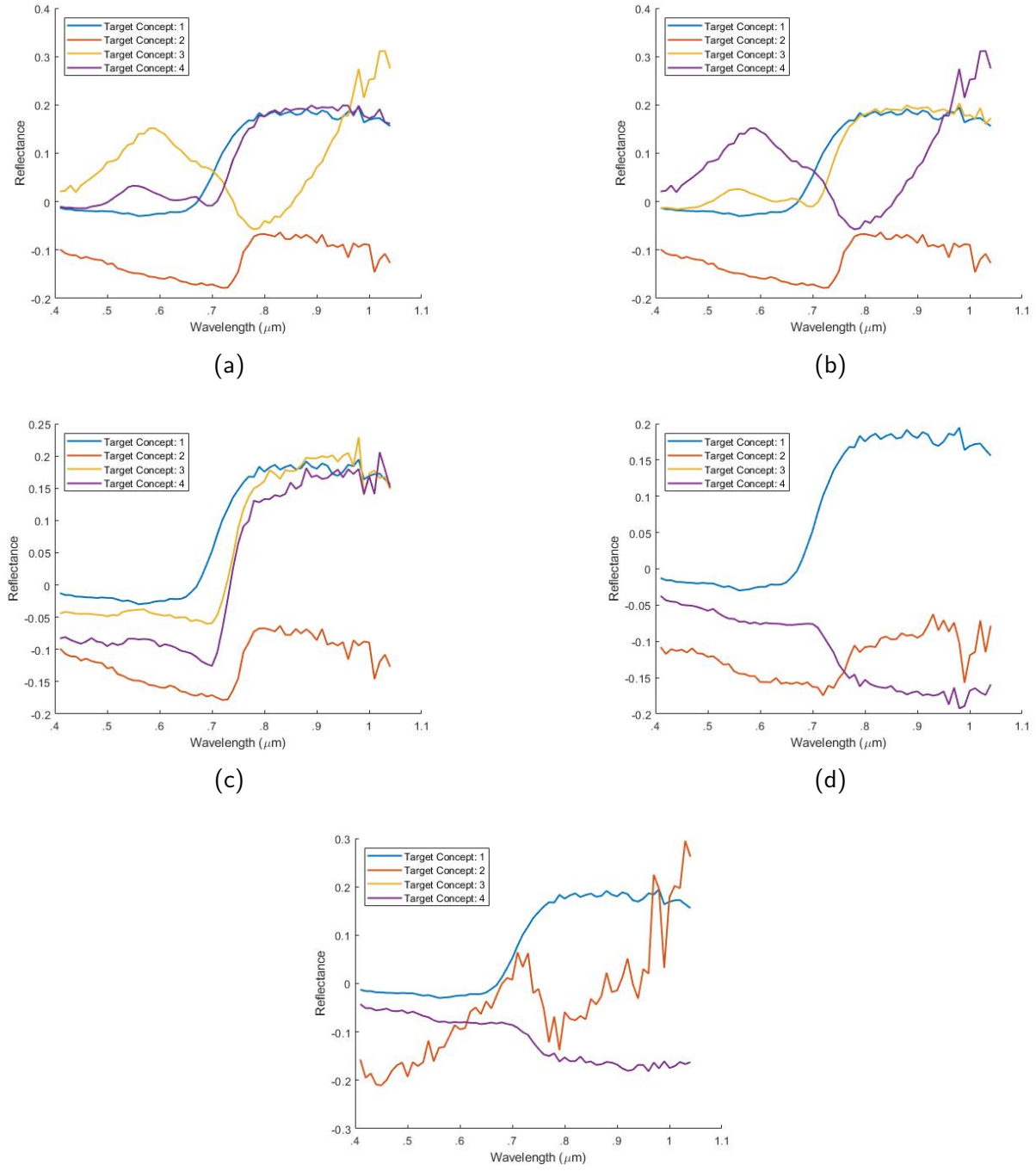


Figure B-9. Initialized signatures for Multi-Target MI-ACE using original multi-target objective function with various uniqueness term weights, α, on MUUFL Gulfport data, all targets. (a) $\alpha = 0.05$. (b) $\alpha = 0.1$. (c) $\alpha = 0.2$. (d) $\alpha = 0.5$. (e) $\alpha = 1$.

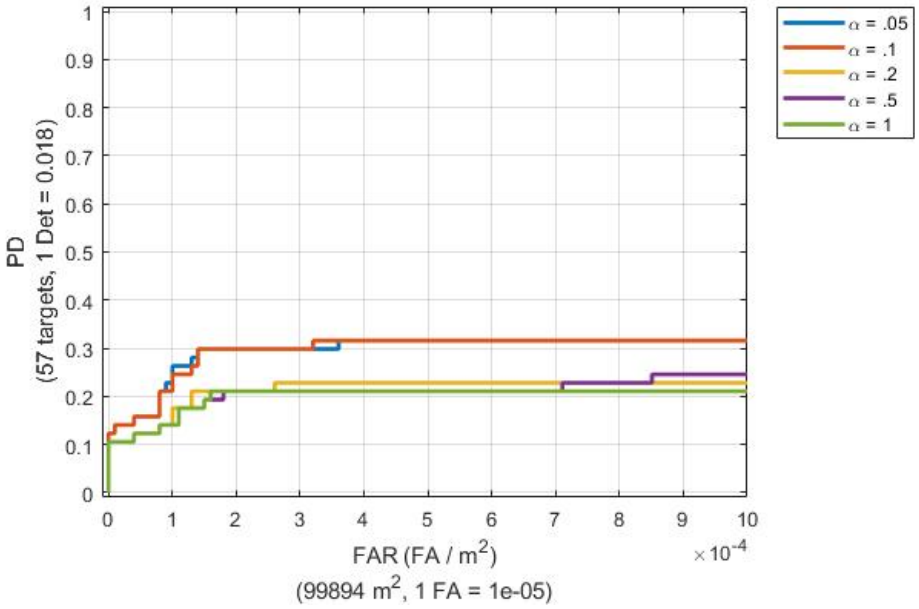


Figure B-10. MUUFL Gulfport ROC curves for initialized signatures with various α settings.

Table B-3. Uniqueness term hyperparameter weight initialization experiment. Showing results using initialized signatures with run time and AUC NAUC for learning four target signatures on all target types.

Uniqueness term	Time	NAUC
$\alpha = 0.05$	6.92	0.295
$\alpha = 0.1$	6.92	0.294
$\alpha = 0.2$	6.89	0.213
$\alpha = 0.5$	6.93	0.206
$\alpha = 1$	6.91	0.199

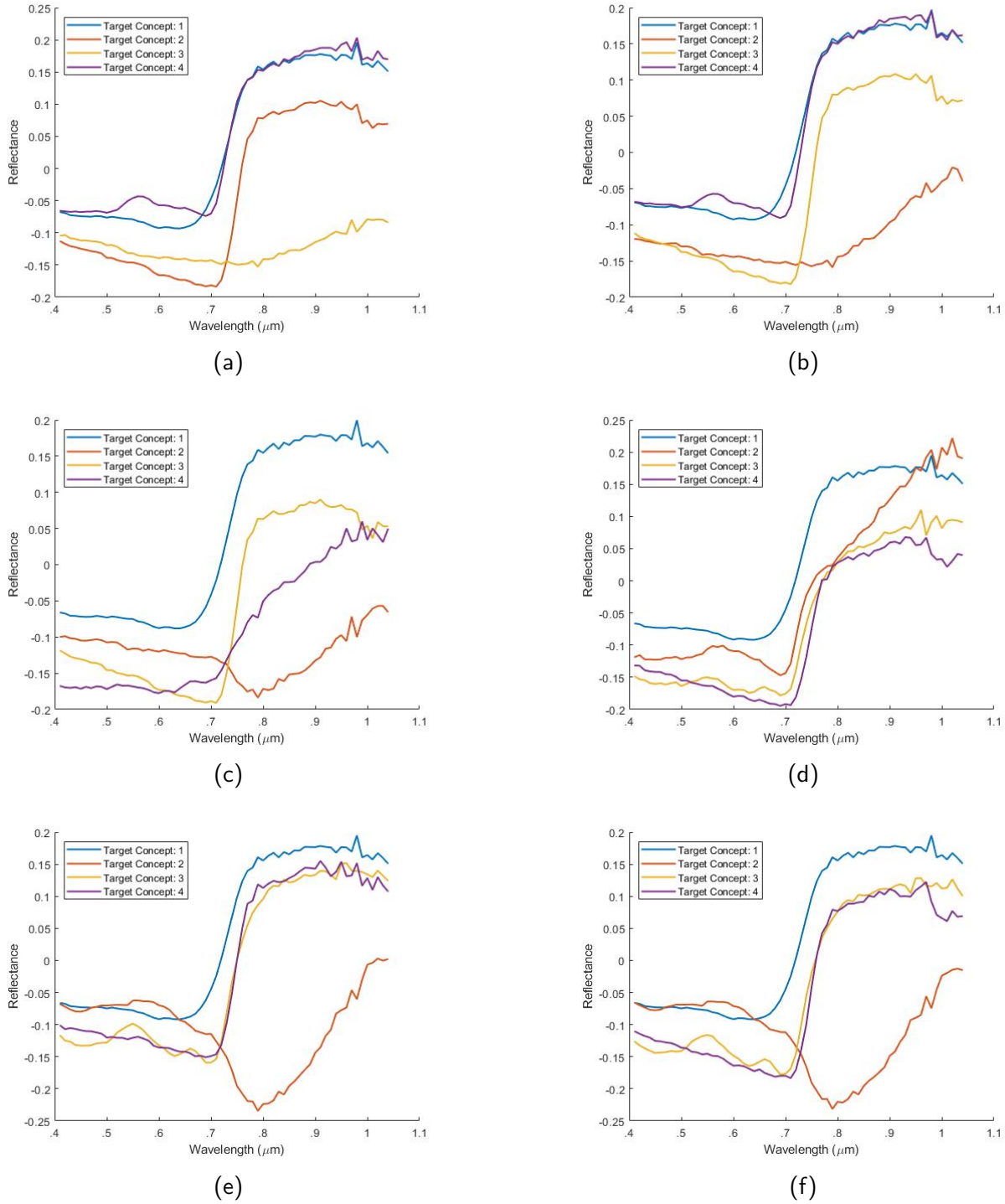


Figure B-11. Initialized signatures for Multi-Target MI-ACE using original multi-target objective function with various number of clusters, C, on MUUFL Gulfport data, all targets. (a) $C = 5$. (b) $C = 6$. (c) $C = 8$. (d) $C = 10$. (e) $C = 15$. (f) $C = 20$.

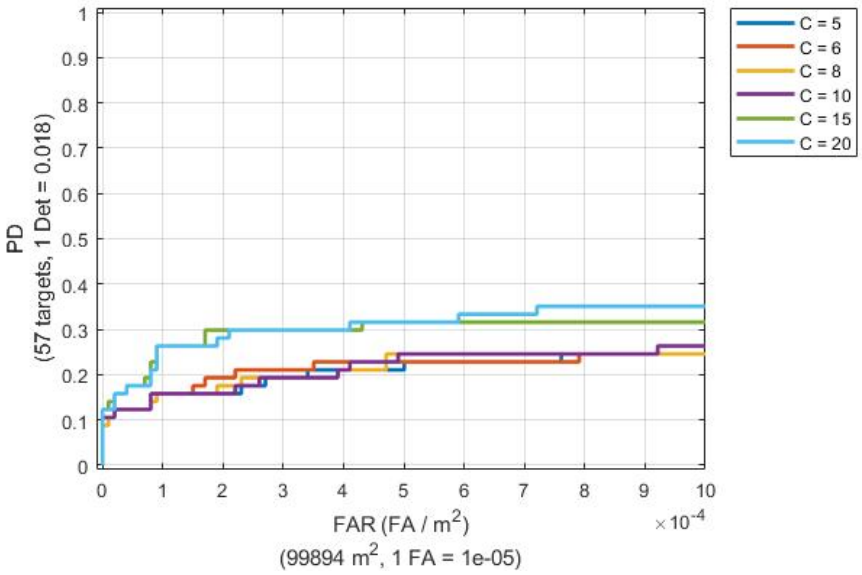


Figure B-12. ROC Curve for Multi-Target MI-ACE using original multi-target objective function with various number of clusters, C, on MUUFL Gulfport data, all targets.

Table B-4. Number of clusters hyperparameter initialization experiment with K-Means. Showing results using initialized signatures with run time and AUC NAUC for learning four target signatures on all target types.

Number of clusters	Time	NAUC
$C = 5$	0.082	0.206
$C = 6$	0.024	0.213
$C = 8$	0.027	0.211
$C = 10$	0.033	0.213
$C = 15$	0.033	0.294
$C = 20$	0.043	0.305

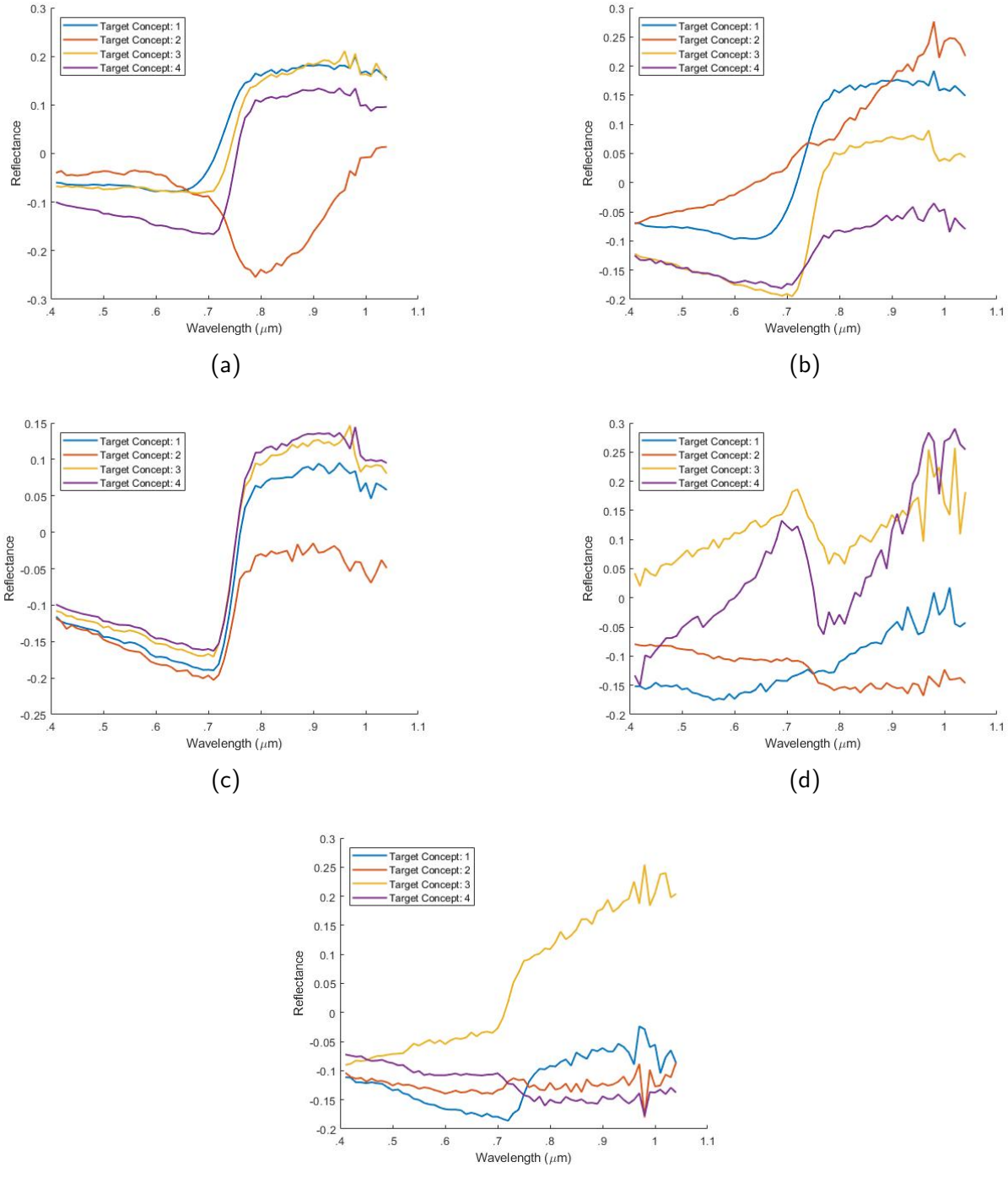


Figure B-13. Initialized signatures for Multi-Target MI-ACE using original multi-target objective function with various cluster rank weights, $(wB+, wD+, wD-)$, on MUUFL Gulfport data, all targets. (a) $(0, 0, 1)$. (b) $(.1, .1, .8)$. (c) $(.3, .3, .3)$. (d) $(.8, .1, .1)$. (e) $(.5, .5, 0)$.

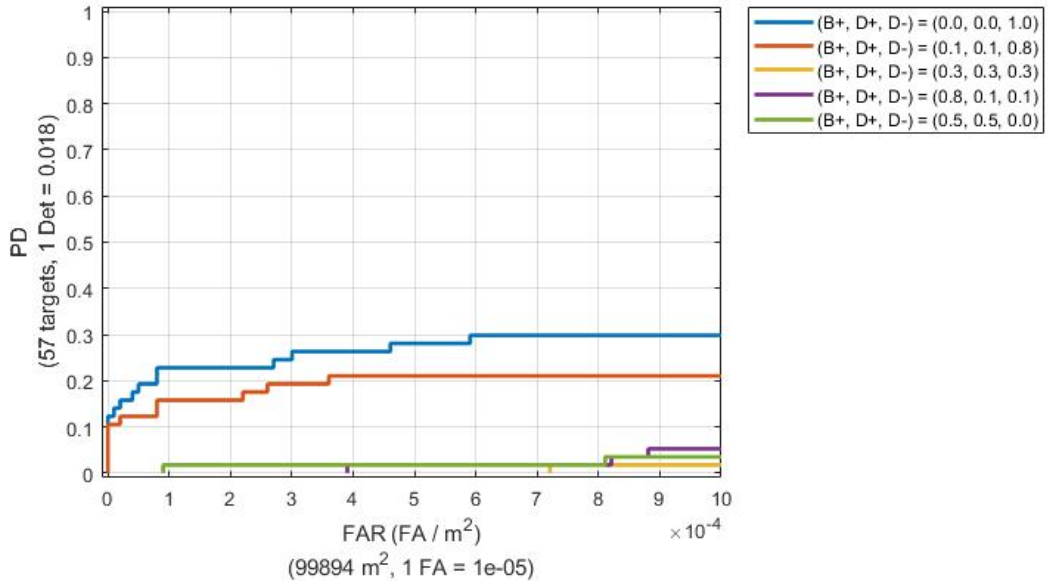


Figure B-14. ROC curves for initialized signatures with various cluster rank weights, $(wB+, wD+, wD-)$, settings.

Table B-5. Cluster rank hyperparameter weights initialization experiment. Showing results using initialized signatures with run time and AUC NAUC for learning four target signatures on all target types.

Weights: (wB+, wD+, wD-)	Time	NAUC
(0, 0, 1)	0.239	0.265
(.1, .1, .8)	0.061	0.193
(.3, .3, .3)	0.060	0.005
(.8, .1, .1)	0.073	0.016
(.5, .5, 0)	0.070	0.019

B.2.2 Optimization Hyperparameter Experiments

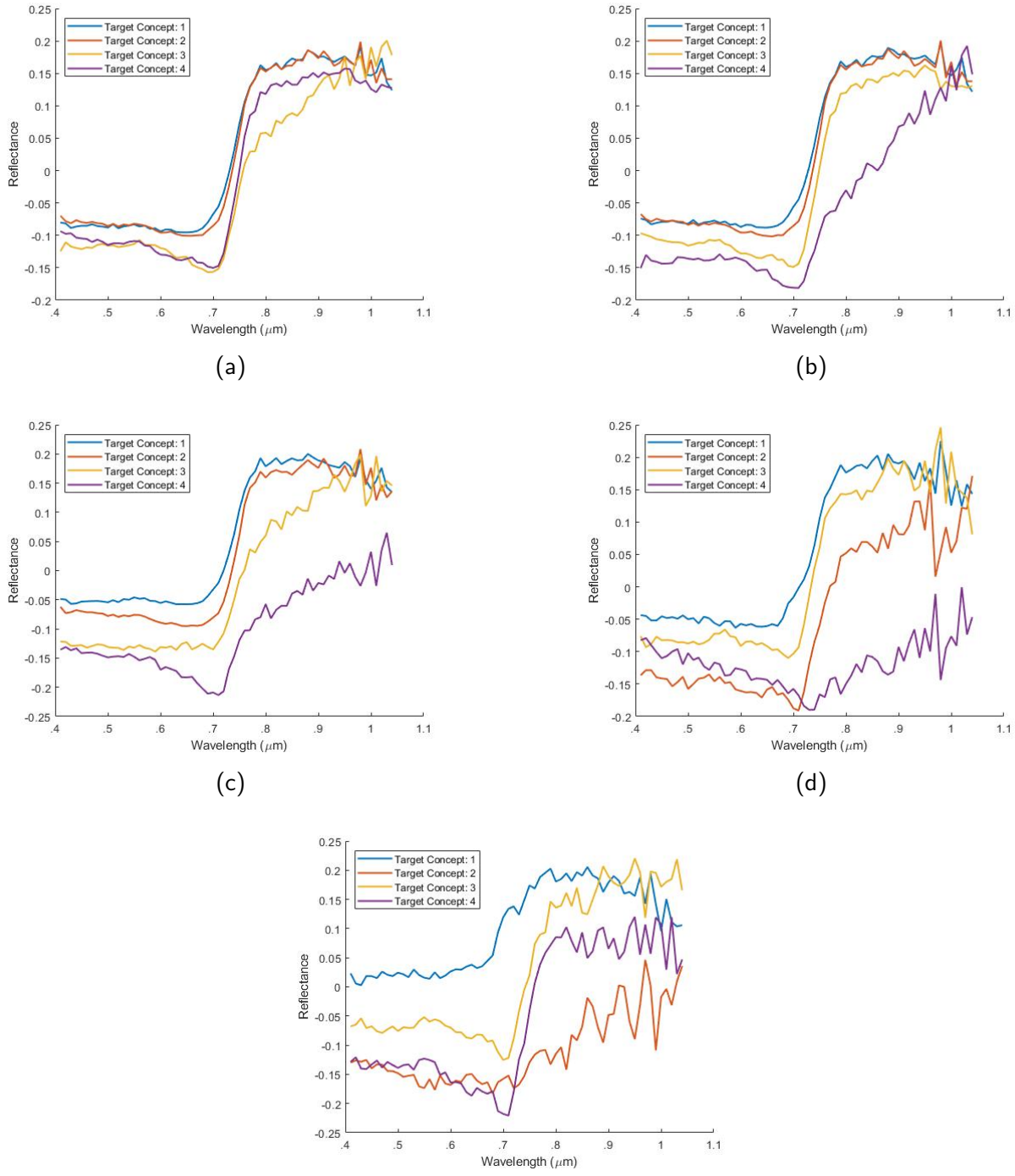


Figure B-15. Optimized signatures for Multi-Target MI-ACE using original multi-target objective function with various uniqueness term weights for optimization, α, on MUUFL Gulfport data, all targets. (a) $\alpha = 0.05$. (b) $\alpha = 0.1$. (c) $\alpha = 0.2$. (d) $\alpha = 0.5$. (e) $\alpha = 1$.

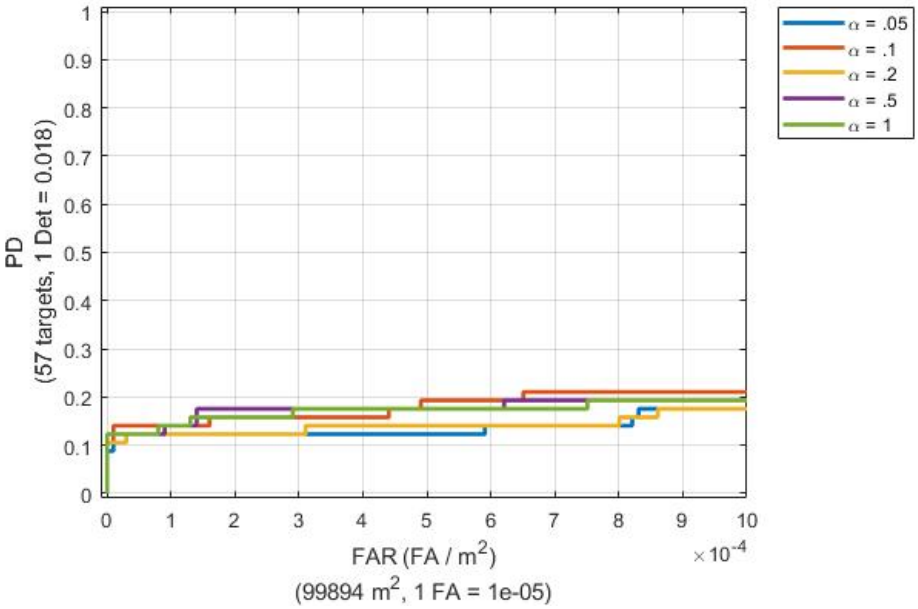


Figure B-16. ROC curves for optimized signatures with various α hyperparameter settings.

Table B-6. Uniqueness term hyperparameter weight optimization experiment. Showing results using optimized signatures with AUC NAUC for learning four target signatures on all target types.

Uniqueness term	NAUC
$\alpha = .05$	0.135
$\alpha = .1$	0.180
$\alpha = .2$	0.140
$\alpha = .5$	0.176
$\alpha = 1$	0.171

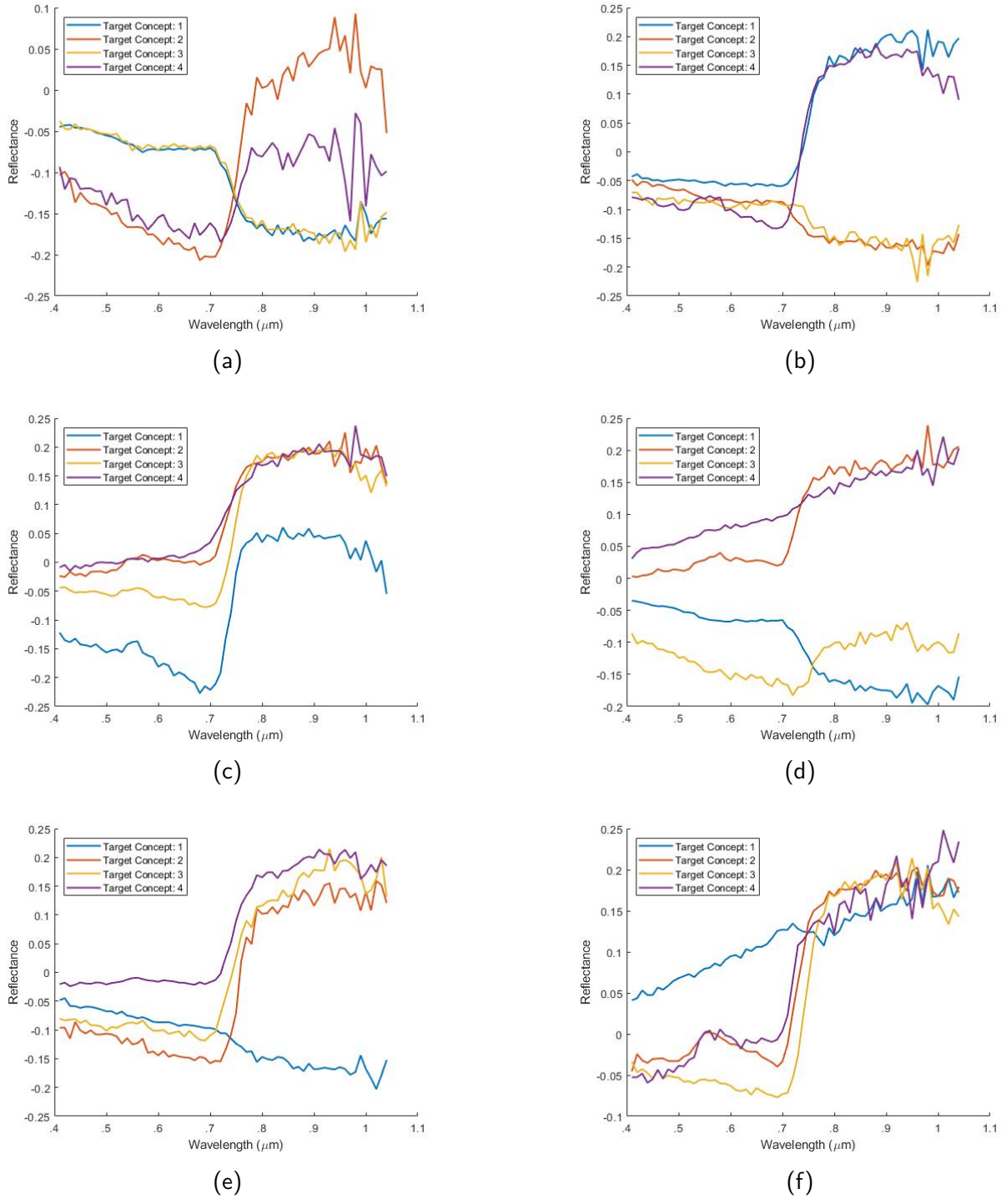


Figure B-17. Optimized signatures for Multi-Target MI-ACE using original multi-target objective function with various kernel bandwidths, σ, for weighted kernel optimization on MUUFL Gulfport data, all targets. (a) $\sigma = 0.05$. (b) $\sigma = 0.1$. (c) $\sigma = 0.2$. (d) $\sigma = 0.4$. (e) $\sigma = 0.6$. (f) $\sigma = 1$.

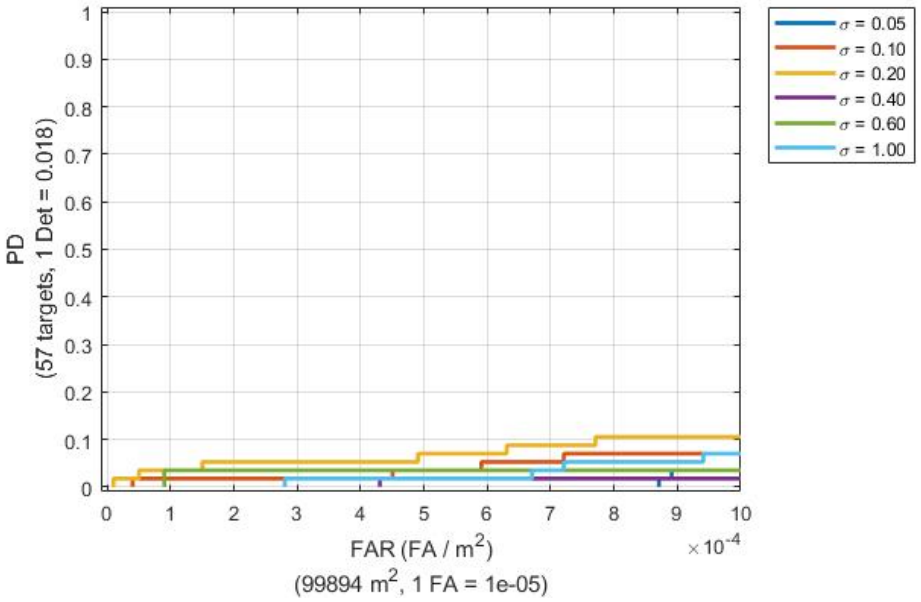


Figure B-18. ROC Curve for Multi-Target MI-ACE using original multi-target objective function with various kernel bandwidths, σ, for weighted optimization on MUUFL Gulfport data, all targets.

Table B-7. Kernel bandwidth hyperparameter optimization experiment for weighted optimization. Showing results using optimized signatures with AUC NAUC for learning four target signatures on all target types.

Bandwidth	NAUC
$\sigma = .05$	0.004
$\sigma = .1$	0.039
$\sigma = .2$	0.068
$\sigma = .4$	0.010
$\sigma = .6$	0.032
$\sigma = 1$	0.024

REFERENCES

Baldridge, AM, Hook, SJ, Grove, CI, and Rivera, G. "The ASTER spectral library version 2.0." *Remote Sensing of Environment* 113 (2009).4: 711–715.

Basener, William F. "Clutter and anomaly removal for enhanced target detection." *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XVI*. vol. 7695. International Society for Optics and Photonics, 2010, 769525.

Broadwater, Joshua and Chellappa, Rama. "Hybrid detectors for subpixel targets." *IEEE transactions on pattern analysis and machine intelligence* 29 (2007).11: 1891–1903.

Chen, Scott Shaobing, Donoho, David L, and Saunders, Michael A. "Atomic decomposition by basis pursuit." *SIAM review* 43 (2001).1: 129–159.

Dietterich, Thomas G, Lathrop, Richard H, and Lozano-Pérez, Tomás. "Solving the multiple instance problem with axis-parallel rectangles." *Artificial intelligence* 89 (1997).1-2: 31–71.

Gader, P, Zare, A, Close, R, Aitken, J, and Tuell, G. "Muufl gulfport hyperspectral and lidar airborne data set." *Univ. Florida, Gainesville, FL, USA, Tech. Rep. REP-2013-570* (2013).

Glenn, T, Zare, A, Gader, P, and Dranishnikov, D. "Bullwinkle: Scoring code for sub-pixel targets (version 1.0)[software]." 2013.

Jiao, Changzhe. *Target concept learning from ambiguously labeled data*. Ph.D. thesis, University of Missouri, 2017.

Jiao, Changzhe, Chen, Chao, McGarvey, Ronald G, Bohlman, Stephanie, Jiao, Licheng, and Zare, Alina. "Multiple instance hybrid estimator for hyperspectral target characterization and sub-pixel target detection." *ISPRS Journal of Photogrammetry and Remote Sensing* 146 (2018): 235–250.

Jiao, Changzhe and Zare, Alina. "Functions of multiple instances for learning target signatures." *IEEE Transactions on Geoscience and Remote Sensing* 53 (2015).8: 4670–4686.

———. "FUMI." https://github.com/GatorSense/FUMI, 2016.

———. "Multiple instance hybrid estimator for learning target signatures." *arXiv preprint arXiv:1701.02258* (2017).

Karem, Andrew and Frigui, Hichem. "Fuzzy clustering of multiple instance data." *Fuzzy Systems (FUZZ-IEEE), 2015 IEEE International Conference on*. IEEE, 2015, 1–7.

———. "Multiple Instance Learning with multiple positive and negative target concepts." *Pattern Recognition (ICPR), 2016 23rd International Conference on*. IEEE, 2016, 474–479.

Kraut, Shawn and Scharf, Louis L. "The CFAR adaptive subspace detector is a scale-invariant GLRT." *IEEE Transactions on Signal Processing* 47 (1999).9: 2538–2541.

Kraut, Shawn, Scharf, Louis L, and McWhorter, L Todd. "Adaptive subspace detectors." *IEEE Transactions on signal processing* 49 (2001).1: 1–16.

MacQueen, James et al. "Some methods for classification and analysis of multivariate observations." *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. vol. 1. Oakland, CA, USA, 1967, 281–297.

Maron, Oded and Lozano-Pérez, Tomás. "A framework for multiple-instance learning." *Advances in neural information processing systems*. 1998, 570–576.

Maron, Oded and Ratan, Aparna Lakshmi. "Multiple-Instance Learning for Natural Scene Classification." *ICML*. vol. 98. 1998, 341–349.

Murphy, Kevin P. *Machine learning: a probabilistic perspective*. Cram101, 2013.

Nasrabadi, Nasser M. "Regularized spectral matched filter for target recognition in hyperspectral imagery." *IEEE Signal Processing Letters* 15 (2008): 317–320.

Peizhuang, Wang. "Pattern recognition with fuzzy objective function algorithms (James C. Bezdek)." *SIAM Review* 25 (1983).3: 442.

Srinivas, Sampath. "A generalization of the noisy-or model." *Proceedings of the Ninth international conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1993, 208–215.

Tibshirani, Robert. "Regression shrinkage and selection via the lasso." *Journal of the Royal Statistical Society. Series B (Methodological)* (1996): 267–288.

Trabelsi, Mohamed and Frigui, Hichem. "Fuzzy and Possibilistic Clustering for Multiple Instance Linear Regression." *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2018, 1–7.

Wagstaff, Kiri L, Lane, Terran, and Roper, Alex. "Multiple-instance regression with structured data." *Data Mining Workshops, 2008. ICDMW'08. IEEE International Conference on*. IEEE, 2008, 291–300.

Zare, Alina and Gader, Paul. "Sparsity promoting iterated constrained endmember detection in hyperspectral imagery." *IEEE Geoscience and Remote Sensing Letters* 4 (2007).3: 446–450.

———. "Pattern recognition using functions of multiple instances." *Pattern Recognition (ICPR), 2010 20th International Conference on*. IEEE, 2010, 1092–1095.

Zare, Alina, Jiao, Changzhe, and Glenn, Taylor. "Discriminative multiple instance hyperspectral target characterization." *IEEE transactions on pattern analysis and machine intelligence* 40 (2018).10: 2342–2354.

Zhang, Qi and Goldman, Sally A. "EM-DD: An improved multiple-instance learning technique." *Advances in neural information processing systems*. 2002, 1073–1080.

BIOGRAPHICAL SKETCH

James Bocinsky was born in March 1995, in Melbourne, Florida. He received his bachelor's degree in computer engineering from the University of Florida, Gainesville, Florida, in 2017. He continued his studies at the University of Florida in the Department of Electrical and Computer Engineering as a Master of Science student. His research focuses on developing multiple instance learning algorithms for subsurface explosive hazard detection. His other areas of interest include digital signal processing, embedded systems programming, remote sensing, and pattern recognition.