Classification using active polarimetry

Israel J. Vaughn[1,2], Brian G. Hoover[2], and J. Scott Tyo[1]

[1]Advanced Sensing Lab, College of Optical Sciences, University of Arizona
[2]Advanced Optical Technologies, Albuquerque, NM

ABSTRACT

Active (Mueller matrix) remote sensing is an under-utilized technique for material discrimination and classification. A full Mueller matrix instrument returns more information than a passive (Stokes) polarimeter; Mueller polarimeters measure depolarization and other linear transformations that materials impart on incident Stokes vectors, which passive polarimeters cannot measure. This increase in information therefore allows for better classification of materials (in general). Ideally, material classification over the entire polarized BRDF is desired, but sets of Mueller matrices for different materials are generally not separable by a linear classifier over elevation and azimuthal target angles. We apply non-linear support vector machines (SVM) to classify materials over BRDF (all relevant angles) and show variations in receiver operator characteristic curves with scene composition and number of Mueller matrix channels in the observation.

1. INTRODUCTION

Mueller matrices measured at optical wavelengths have not often been utilized to classify material types. Jones *et al*[1] applied SVM to measured Mueller matrices, but their approach was limited; they used phenomenologically derived features as inputs into the support vector machine instead of the full Mueller matrix. Hoover and Tyo[2] applied PCA to attempt invariance, and used linear combinations of Mueller matrix elements selected via an *ad hoc* process. Recently Goudail and Tyo[3,4] published work on maximizing contrast depending on the depolarizing properties of the Mueller matrix, and Anna *et al*[5,6] used linear discriminant analysis (LDA) for classification of materials from Mueller matrices. Anna's LDA paper confirms work done by Hoover and Tyo in 2007[2] using principal components analysis (PCA) to discriminate materials via Mueller matrices of those materials. Weinreb *et al* also applied LDA to laser polarimetric data for glaucoma classification in 1998, but only retardance was measured.[7] LDA and PCA, however, fail to classify materials well over parameters like pose angle and other BRDF variations. We apply non-linear SVM to optical Mueller data to classify materials *over these nonlinear parameters*. When a subset of Mueller matrix channels is used for input into SVM, we use a modified version of Hall's correlation feature selection[8] to select the best subset. We define Mueller matrix channels as *linear combinations of specific Mueller matrix elements*.

Figure 1 shows Mueller data for several materials taken for various azimuthal angles in a (approximately) mono-static measurement configuration, then processed by the PCA algorithm described by Hoover and Tyo.[2] These materials don't cluster in a Gaussian distribution, they have a distribution that can be approximated by a family of splines or polynomials. This example visually indicates the need for a nonlinear classification scheme to discriminate materials in general.

Work has also been done on classification of data from polarized synthetic aperture radar (SAR) instruments.[9–14] Classification of polarimetric SAR data has typically utilized an unsupervised approach, based on phenomenological arguments for the classes (aka clusters). Pierce *et al*[12] pursued a knowledge based classification method which was a hybrid of experimental and phenomenological results, i.e., a semi-supervised approach. Pierce *et al* had good accuracy for their classes, but did not provide a receiver operating characteristic (ROC) curve.

Further author information: email : ivaughn@optics.arizona.edu
http://www.optics.arizona.edu/asl

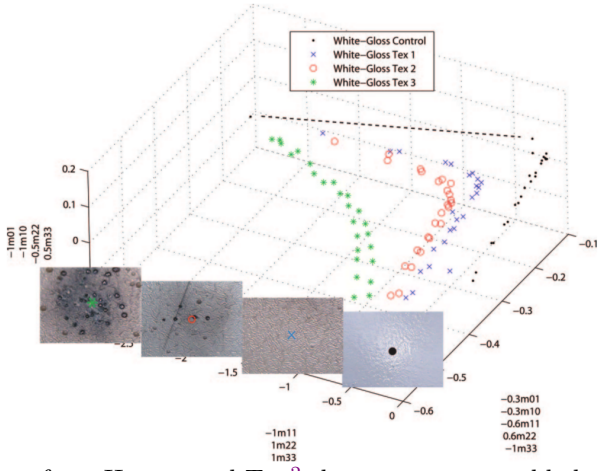


Figure 1: An example of clusters from Hoover and Tyo[2] that are not separable by a linear classifier. Notice that a parameter of curves can be fitted to the data in this case.

2. MUELLER MATRIX DATABASE

The database consists of 41,459 measurements of 47 different materials, measured with a previously described laser polarimeter[2,15] at a wavelength of $\lambda = 1064nm$. The measurement set of each material consists of measurements of different locations of particular samples, different BRDF angles, and different samples of particular materials. The database contains some targets of interest (TOIs), some clutter such as vegetation, and backgrounds such as sand and soils. This analysis is per pixel, and does not take into account spatial correlations that would be present in an imaging system.

Three datasets are analyzed here; A,B, and C. Dataset A is the raw dataset, with each material with each material uniformly distributed, this results in the clutter being over represented as a percentage of the total. Dataset B was also redistributed so that background and clutter filled 50% of the dataset and the TOIs filled 50% of the dataset. In the TOI subset of the B dataset, the TOIs were uniformly distributed (over material type), and in the background and clutter subset, the forty background and clutter materials were uniformly distributed (over material type). This was needed for an empirical risk minimization metric (ERM), similar to accuracy, to be applied to support vector machine model (SVM) selection. Dataset C was distributed such that most of the Mueller matrices were background. The datasets are described with more detail in table 1 below.

Dataset Name	Total MMs	% Backgrounds	% Clutter	% TOIs	AWGN	Comments
A	252,321	10	76	14	1%	
B	664,517	3.5	46.5	50	1%	50-50 distribution between TOIs and non-TOIs. TOIs were mixed with background.
C	1,495,570	77.5	20	2.5	0.5%	Small percentage of TOIs. TOIs were mixed with background.

Table 1: Dataset parameters for A, B, and C datasets.

The number of measurements per material is not evenly distributed, so each material was replicated the number of times needed to have equal numbers of Mueller matrices for each material type for each subset. This was accomplished by first selecting the total number of Mueller matrices needed, say N for each material in a subset. Then if a material type had fewer than N measured Mueller matrices in the data set, Mueller matrices from the existing material type were selected at random, propagated through an ideal dual rotating retarder

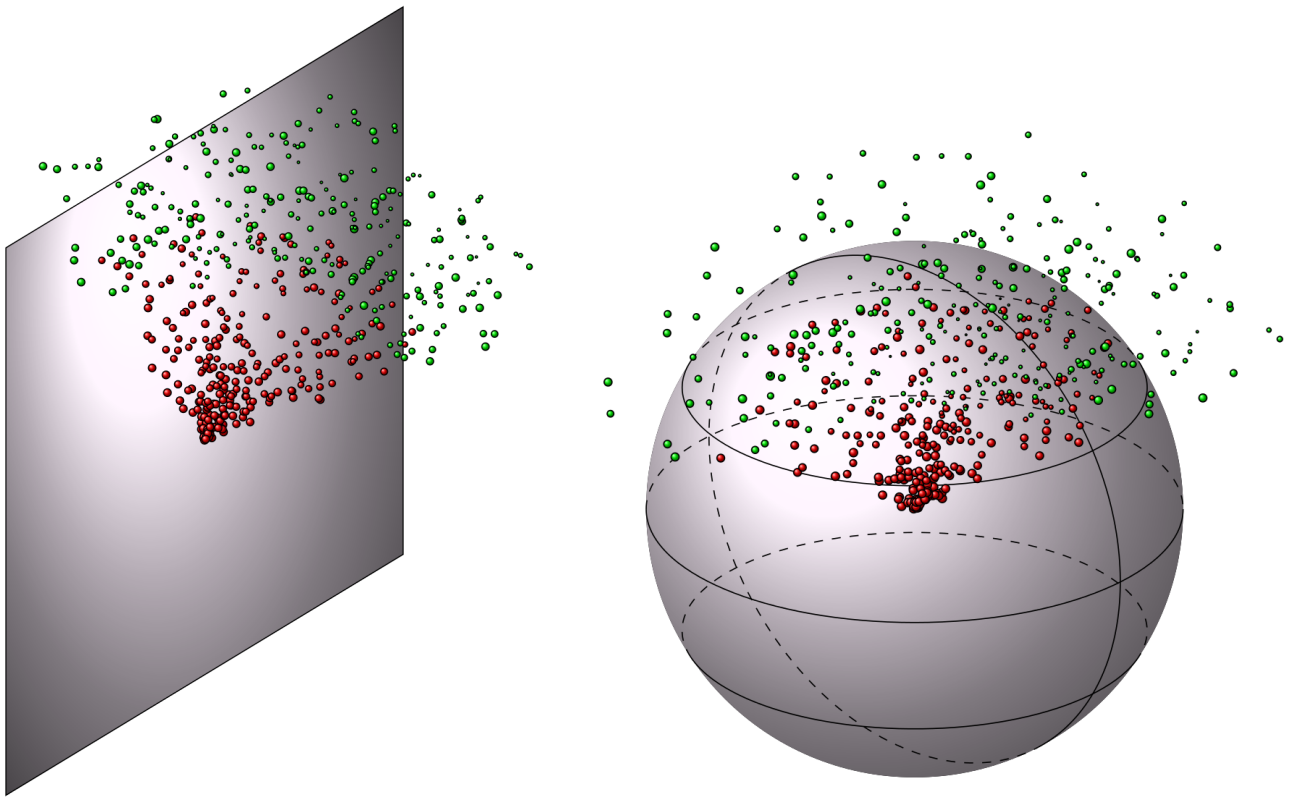
polarimeter equation,[15–19] white Gaussian noise (AWGN) added to the irradiance matrix (at a certain fraction of the mean), then re-inverted back to the Mueller matrix until there were N Mueller matrices of each material type. If there were more than N Mueller matrices of a particular material type, then N of them were chosen at random. These equally distributed datasets are utilized in the following analysis.

The TOIs were mostly measured without background present. This is an unrealistic scenario, so for the B and C datasets, background was mixed with the TOI data. Background Mueller matrices were randomly selected from the background subset, then added in a certain proportion of the mean irradiance values to each TOI Mueller matrix that was not measured with background. The proportions ranged from 1:31 to 1:4, TOI to background. This mixing was linear, and assuming that we have linear Mueller matrices, and that backgrounds are Lambertian, then adding background linearly in irradiance is valid. The C dataset had a reduced number of materials in the clutter subset.

3. METHODS

3.1 Support vector machines

Support vector machines,[20–22] a nonlinear learning/classification method, are used in this study, and we explain them briefly here. Conceptually support vector machines are simple. Suppose we have data points in some n dimensional space, each one assigned to a class in the set $\{+1, -1\}$. Now suppose that we want to construct an $n-1$ dimensional surface (called a manifold) that separates the data in the two classes as best as possible. The first surface to try is just an $n-1$ dimensional plane. The plane is flat, so will often be a poor separating surface (if for example a spherical surface better separates the classes).



(a) Hyperplane separation (b) Sphere separation

Figure 2: A set of points for two different classes in three dimensions, (●) is the -1 class and (●) is the $+1$ class. Note that for this example the sphere is the best possible surface which separates the two classes, while a plane can never separate the classes very well, no matter the orientation of that plane.

Figure 2 demonstrates this issue, notice that with the data displayed no plane can separate the two classes, only a spherical surface (or something similar) can separate the classes.

It may be difficult to derive a surface that separates data classes, especially in spaces with greater than three dimensions as visualization is difficult. Support vector machines (SVMs) are one technique with which to derive a suitable surface. SVMs construct this surface in the data space by employing a kernel function, which maps an n dimensional point into an (usually) infinite dimensional space. This is denoted the dual space. Essentially points can almost always be separated in the infinite dimensional space, and the inverse map of that separation surface generates a complicated surface in the data space. The "support" part of support vector machines comes from the fact that SVMs use the vectors themselves to define the surface. We will not delve into details about how SVMs are constructed or why they are mathematically valid, see Cortes[20] for the SVM implementation (or Burges[21] for a detailed analysis) and Lax[23] or Stakgold[24] for the general mathematical functional analysis details.

Note that algorithms like principal components analysis (PCA), linear discriminant analysis (LDA), and empirical orthogonal function (EOF) are all examples of algorithms that can only separate data with a plane as shown in Fig. 2a. These are all examples of *linear* classifiers. If a finite set of parameters is being estimated (i.e. more than two classes but not an infinite number of classes) the most general linear estimator is called the *Wiener estimator* which includes statistical classification. Matched filters and LDA are special cases of the Wiener estimator. PCA and EOF are similar to the prewhitening step for prewhitened matched filters.[25]

3.2 SVM parameters

First, there are two ways to compute SVMs. SVMs can be reduced to a quadratic programming problem :

$$\frac{1}{2}\left\|\sum_{i=1}\alpha_i\phi(\mathbf{x}_i)\right\|^2 - \sum_i y_i\alpha_i \text{ s.t.} \begin{cases} \sum_i \alpha_i = 0 \\ 0 \le y_i\alpha_i \le C \end{cases} \tag{3.1}$$

the above expression is minimized over $\boldsymbol{\alpha}$, where the inner product $\big(\phi(\mathbf{x}_i),\phi(\mathbf{x}_j)\big)$ is the kernel function, the $y_i \in \{-1,1\}$s are the labels, the $\mathbf{x}_i$s are the data vectors or points, and C is the box constraint. Note that this minimization is done in the dual (or kernel) space rather than in the data space. SVM constructs a surface which maximizes the distance between the two classes, this is denoted the *maximal margin hyperplane*, and this hyperplane is in the dual or kernel space. Note that the hyperplane corresponds to some *non-planar* surface (manifold) in the data space.

The two parameters that can then be adjusted are the box-constraint, C, and the kernel function, $\big(\phi(\mathbf{x}_i),\phi(\mathbf{x}_j)\big)$. Nearly always, given enough time, SVM training can find a surface in an infinite dimensional kernel space which completely separates two classes. This, however, often leads to poor performance on subsequent data (which was not trained on originally with SVM) because data always has statistical errors and SVM is a non-probabilistic learning machine. Basically, SVM finds a surface which works perfectly for the given data, but is too specific, and results in many mis-classifications on subsequent data. This is called over-training, and is a result of a low number of training samples not properly representing the underlying true probability distribution.

The C parameter basically determines the penalty for mis-classified data points during the SVM training phase. If C is small, then the penalty is small, and many points can be mis-classified, if C is large then fewer points will be mis-classified, but the margin distance may shrink. This is equivalent to a kind of regularization parameter. This description is not complete, as we have not discussed how the *soft margin* parameter enables the mis-classification. For details, see Cortes and Vapnik.[20]

The other parameter is the kernel function. This function determines the kinds of shapes that are generated in the data space. In real world use, most people have good results using the Gaussian kernel :

$$\big(\phi(\mathbf{x}_i),\phi(\mathbf{x}_j)\big) = e^{-\|\mathbf{x}_i-\mathbf{x}_j\|^2/2\sigma^2} = e^{-(\|\mathbf{x}_i\|^2+\|\mathbf{x}_j\|^2-2\mathbf{x}_i\cdot\mathbf{x}_j)/2\sigma^2} \tag{3.2}$$

also commonly denoted a radial basis function (RBF) in the literature. The Gaussian is parameterized with σ alone, which makes it easy to characterize. Another kernel often used is the polynomial kernel :

$$k(\mathbf{x}_i,\mathbf{x}_j) = \Big[a\mathbf{x}_i\cdot\mathbf{x}_j\Big]^p \tag{3.3}$$

where p is the degree of the polynomial. The parameter a and p can be varied. These parameters are often searched over to generate a reasonable or best-possible SVM for classification later on. We have utilized the Gaussian kernel on datasets A and C and polynomial kernel on dataset B.

Note that the input into a trained SVM is a set of vectors or points with some dimension, n. The SVM then maps each input vector to some number. When SVM is trained, it attempts to map points to -1 or $+1$ using a decision surface, but in general the output is a real number, i.e., when an arbitrary point is input into the trained SVM, the output is some real number denoted the *test statistic*, λ. To classify a point, some threshold is defined, c_t, and when $\lambda > c_t$ the point is considered to be in class $+1$ and when $\lambda \leq c_t$ the point is considered to be in class -1. An ROC curve can then be generated by taking a set of points, with known classes, and varying c_t while plotting the *true positive rate* (aka the probability of detection, the sensitivity, the hit rate, etc.) vs the *false positive rate* (aka the false alarm rate, the fall out, 1 - specificity, etc.).

3.3 SVM training and implementation

We trained SVM models using the Gaussian kernel for datasets A and C, and the polynomial kernel for dataset B. The datasets are divided into two classes for SVM evaluation; with TOIs being assigned to the $+1$ class and background and clutter being assigned to the -1 class.

In order to prevent over-training the following procedure was followed for the Gaussian kernel:

(a) The original dataset is separated into two sub-sets, a holdout set and the training set. This is done by a uniformly distributed random assignment. The holdout set is 20% of the original dataset and the training set is 80% of the original training set for A, and the training set is 5% and the holdout set is 95% of the original dataset for C respectively. Dataset C was only trained on 5% of the data due to computational time constraints caused by the size of dataset C.

(b) The training set is subsequently divided into k smaller sets, again by a uniformly distributed random assignment. These are commonly referred to as folds. We have used $k = 10$ as this number generally yields the best results, from the SVM community's empirical experience.

(c) Once the k folds are created, $k-1$ folds are combined into one set for training, while the remaining fold is used for testing the trained SVM (i.e. no points in the remaining fold are used in SVM training), this can be done for each fold so an SVM is trained a total of k times. Each SVM is trained using a single specific parameter vector, i.e., a single point in the parameter space. Then each trained SVM can be used to classify the withheld fold, and some performance measure can be applied to the classified points. The performance measure can then be averaged over the k folds to get a reasonably unbiased performance measure for particular parameters used for the SVM.

(d) The above is done for a set of parameters $\big(\{\sigma_1, \sigma_2, ..., \sigma_m\}, \{C_1, C_2, ..., C_n\}\big)$, and the parameters with the best average performance measure are selected for use. The SVM is then re-trained on the entire training set using the chosen parameters, i.e., over all k folds combined together instead of $k-1$ folds.

(e) The trained SVM from above is then used on the held out data, and the performance measure calculated. This performance measure is the most likely real world performance measure.

For the polynomial kernel the following procedure is followed:

(a) The original dataset is separated into two sub-sets, a holdout set and the training set. This is done by a uniformly distributed random assignment. For B the holdout set is 20% of the original dataset and the training set is 80% of the original training set.

(b) The training set is trained on with a polynomial kernel SVM for specific a, p, and C.

(c) The above is done for a set of parameters $\big(\{a_1, a_2, ..., a_m\}, \{C_1, C_2, ..., C_n\}\big)$, with $p = 3, 4$ and the parameters with the best average performance measure are selected for use. The performance measure for the polynomial case is empirical risk minimization (ERM, explained below).

(d) The trained SVM from above is then used on the 20% of the data held out at the beginning, and the performance measure calculated. This performance measure is the most likely real world performance measure.

3.4 Iterative SVM

The quadratic programming problem posed in section 3.2 uses a Hessian matrix in practice. Suppose that the number of data points in the training set is N, then the size of the Hessian matrix will be N^2. This severely limits the size of the training database, with 8GB of RAM only $\sim 13,000$ sixteen dimensional data points can be used in the training set for the Matlab SVM training implementation.

Most classification algorithms and statistical observers can be recast as an iterative problem. Sequential minimal optimization (SMO) using the Karush-Kuhn-Tucker (KTT) conditions is used to train an SVM iteratively. The iterative implementation introduces three other parameters, the maximum allowed iterations, the tolerance, and the KTT violation parameter. The maximum allowed iteration parameter is self explanatory. The tolerance is the number used to compare the difference between the previous iteration to the current iteration, if the SVM is not changing much then the algorithm can be terminated, i.e., it is the convergence tolerance parameter. The KTT conditions are a set of conditions (inequalities) that many locally linear optimization problems must satisfy, whether they are globally linear or not (see http://www.svms.org/kkt/), also see Smola and Schölkopf[22] for details. Basically the amount of violation of the KTT conditions and the margin of the hyperplane result in a tradeoff. We will probably not be searching over these conditions once reasonable values are found (usually both the tolerance and the KTT violation are small values). Note that setting a maximum iteration value is equivalent to regularizing the optimization problem.

Catanzaro *et al*[26] developed a CUDA codebase for iterative SVM in 2008. CUDA is a C-like language for compiling code to run on Nvidia graphics processing units, which are conducive to data parallel algorithms. We have modified this code slightly to run with the latest CUDA drivers. For the same input data, we realize a 10-50× speedup over the SMO Matlab code. We have used the CUDA implementation for this analysis.

3.5 Performance measures

For statistical two set classification, an ROC curve displays the most general performance information. There does exist a "best" ROC curve, which is the ROC curve produced by the theoretical ideal observer (theoretically optimal classifier), for details see Barrett and Myers.[25] It is provable that no other ROC curve will be higher, and to the left, of the ideal observer's ROC curve. Ideal observers are, however, difficult to find or compute, as they require knowledge of statistical parameters that are usually unknown. In the case where an ideal observer cannot be computed, or if it is not known if a particular classifier is the ideal observer, some performance measure must be used to compare observers (classifiers).

We assumed a scenario where there was sensitivity to the false positive rate, and used a modified area under (the ROC) curve (AUC) metric. Specifically

$$AUC_a = \int_0^a ROC(P_{FA}), \tag{3.4}$$

for some a used as the performance measure. The SVM models were selected using $AUC_{0.1}$ as the performance metric for the A and C datasets.

Another performance measure is empirical risk minimization (ERM). The B models were selected using the following risk bound;[20]

$$R(\alpha) \leq R_{\text{emp}}(\alpha) + \sqrt{\frac{h[ln(2\ell/h)+1] - ln(\eta/4)}{\ell}}, \tag{3.5}$$

where ℓ is the number of observations, h is the VC dimension, and $0 < \eta \leq 1$, and R_{emp} is the empirical risk.[20,21] Furthermore

$$h = \binom{d+p-1}{p}, \tag{3.6}$$

the "n choose k" notation, for a homogeneous polynomial kernel where d is the dimensionality and p is the degree of the polynomial.[21] The empirical risk is :

$$R_{\text{emp}} = \frac{1}{2\ell} \sum_{i=1}^{\ell} |y_i - f(\mathbf{x_i}, \boldsymbol{\alpha})| \tag{3.7}$$

where $f(\mathbf{x_i}, \boldsymbol{\alpha})$ represents an indicator function composed with the function representing the output of some arbitrary SVM, dependent on some parameter vector $\boldsymbol{\alpha}$, and $y_i \in \{-1, +1\}$ is the tag.[21] Note that for the Gaussian kernel, h, the VC dimension, is usually infinite, so ERM is not a good performance metric for Gaussian SVM.[20,21]

3.6 Principal component analysis

A preconditioning step is to apply principal components analysis (PCA). PCA is a linear transformation on the data that orients the directions of basis vectors in the direction of maximum variance of the data. In this case our data are Mueller matrices, which are flattened to sixteen dimensional vectors in order to apply PCA. Since SVM is a nonlinear classifier, PCA has no effect on the SVM performance, unless we want to reduce the data dimensionality, for instance we can take eight channels of the Mueller matrices as the inputs into the SVM algorithm, then PCA can increase performance,[2] along with correlation feature selection, discussed below.

3.7 Correlation feature Selection

A dataset has a natural underlying structure, often corrupted by measurement noise and other uncertainties. When estimating the underlying structure for classification or estimation purposes, the noise and uncertainties can cause overtraining, which is described in the SVM document. There are several methods to reduce this overtraining, one of which is systematic dimensionality reduction. Correlation feature selection (CFS) developed by Hall in 2000 is one such method[8] for a two class dataset.

Hall CFS is a method which maximizes correlation of the features (aka vector element values, in our specific case the Mueller matrix channels) to the classes of those features or vectors. Without going into great detail, Hall CFS is defined by the following equation :

$$M_s = \frac{k\overline{r_{\text{cf}}}}{\sqrt{k + k(k-1)\overline{r_{\text{ff}}}}}, \tag{3.8}$$

where k is the number of features, $\overline{r_{\text{cf}}}$ is the mean correlation between the features and the classes, $\overline{r_{\text{ff}}}$ is the mean intra-feature (i.e. feature-feature) correlation, and M_s is the merit value. M_s is computed for a specific set of k features, i.e., if we have three features originally, then for $k = 2$ the following combinations can be made : $\{1, 2\}, \{1, 3\}, \{2, 3\}$ and $\overline{r_{\text{cf}}}$, $\overline{r_{\text{ff}}}$, and then M_s must be computed for each specific set. The set with the maximum M_s is then selected as the two feature set for dimensionality reduction (feature selection).

Note that given n total features of the underlying dataset, there are $\binom{n}{k}$ subsets of size k. The CFS calculation is applied for every combination of k features taken from the sixteen initial Mueller matrix features. M_s is then sorted by value, and the features resulting in maximum M_s are chosen for selection and subsequent input into the SVM. PCA can be used as a preconditioner to CFS.

4. RESULTS

4.1 Dataset A

The SVM training on the A dataset used 10-fold cross validation for SVM model selection. The parameter space was logarithmically gridded with $0.001 \leq \sigma \leq 10,000$ and $0.01 \leq C \leq 500$. 330 parameter points were tested, requiring 3300 training iterations. Total training time was $\sim$ 40 hours on an Nvidia GTX 580 GPU with 3GB of RAM. The top ten models were selected using the $AUC_{0.1}$ metric averaged over the 10 folds. The top ten models were then used to classify the validation set, shown in figure 3. The results are very good, however the TOIs are not mixed with backgrounds in this case.

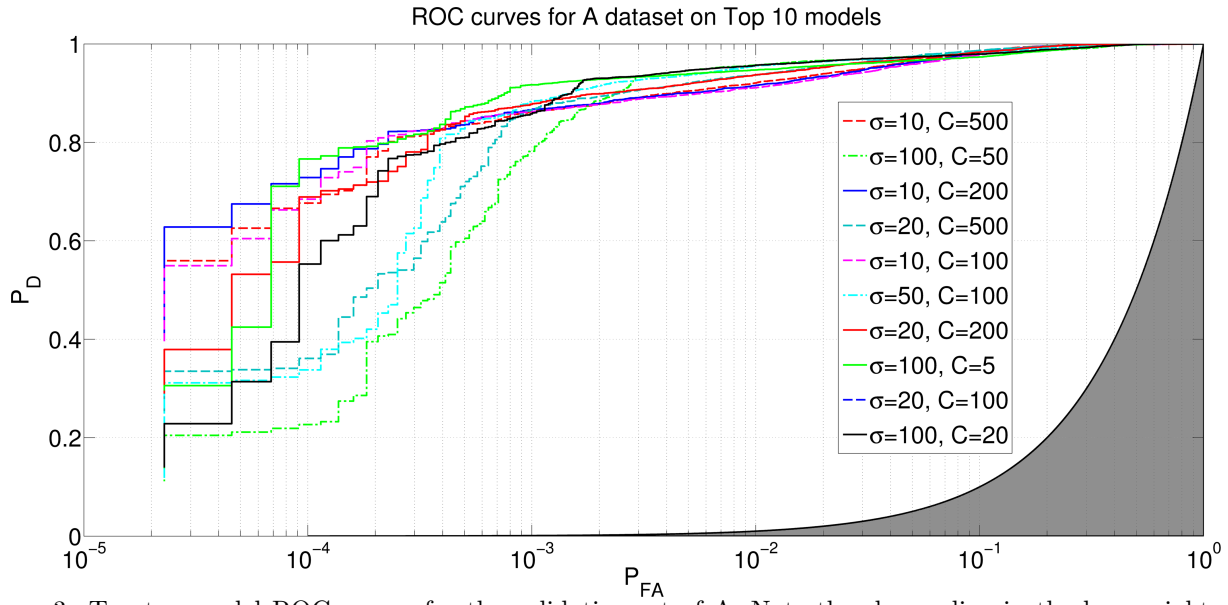


Figure 3: Top ten model ROC curves for the validation set of A. Note the chance line in the lower right and everything below the chance line colored in gray.

4.2 Dataset B

The SVM training for B utilized the polynomial kernel with $p = 3, 4$. The ERM metric is used to evaluate performance.

4.2.1 Polynomial SVM, $p = 3$

For the $p = 3$ case the parameter space was logarithmically gridded with $0.001 \leq a \leq 10$ and $0.01 \leq C \leq 500$. 195 parameter points were tested, requiring 195 training iterations. Total training time was ~ 23 hours on an Nvidia GTX 580 GPU with 3GB of RAM. The top ten models were selected using the ERM metric. The ROC curves are shown in 4

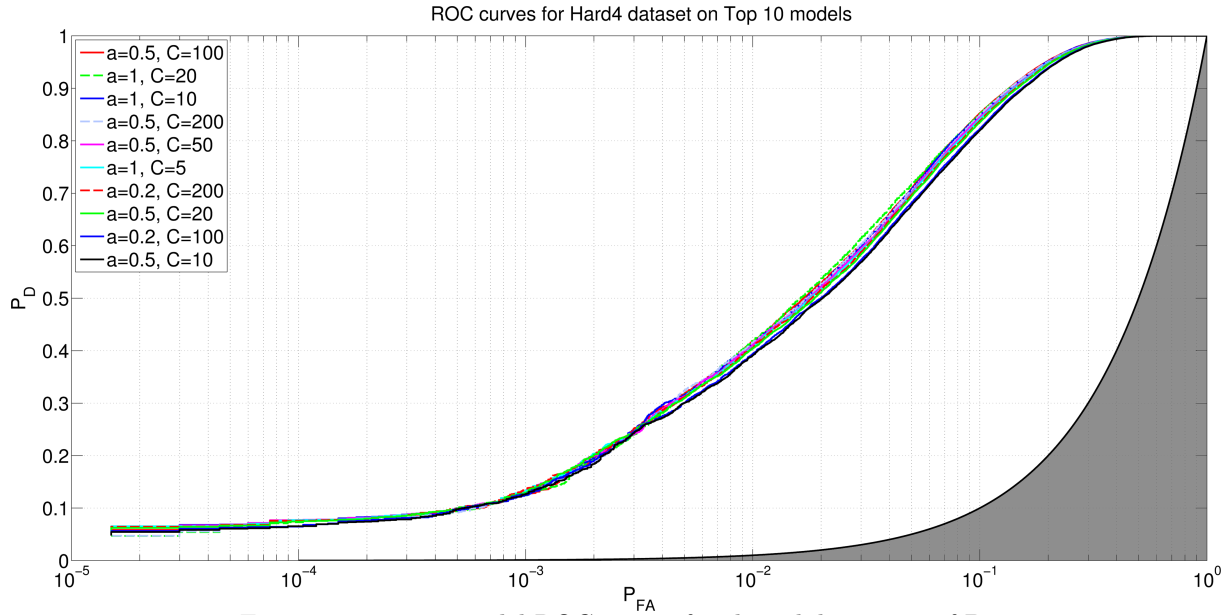


Figure 4: Top ten model ROC curves for the validation set of B.

These ROC curves are not as good as A, but we switched to the polynomial kernel with the ERM performance metric and also have data that is more realistic since the TOIs were mixed with background. In this specific case, with $p = 3$ and $\ell = 531,265$, we have $h = 0.1122$. The top ten ERM bounds are given in the table below :

Model Rank (lowest bound)	Bound
1	0.3854
2	0.3855
3	0.3856
4	0.3860
5	0.3861
6	0.3864
7	0.3868
8	0.3871
9	0.3883
10	0.3888

Table 2: Vapnik's Risk Bound for the B dataset with $p = 3$.

We did not continue the model parameter search due to the extreme time involved; only 195 model evaluations took nearly 23 hours. It appears that the training convergence time is much slower for the polynomial kernel with $p = 3$ than for the Gaussian kernel. We attempted to mitigate the computational performance issue by changing $p = 4$, which should have allowed faster convergence due to more degrees of freedom in the kernel space.

4.2.2 Polynomial SVM, $p = 4$

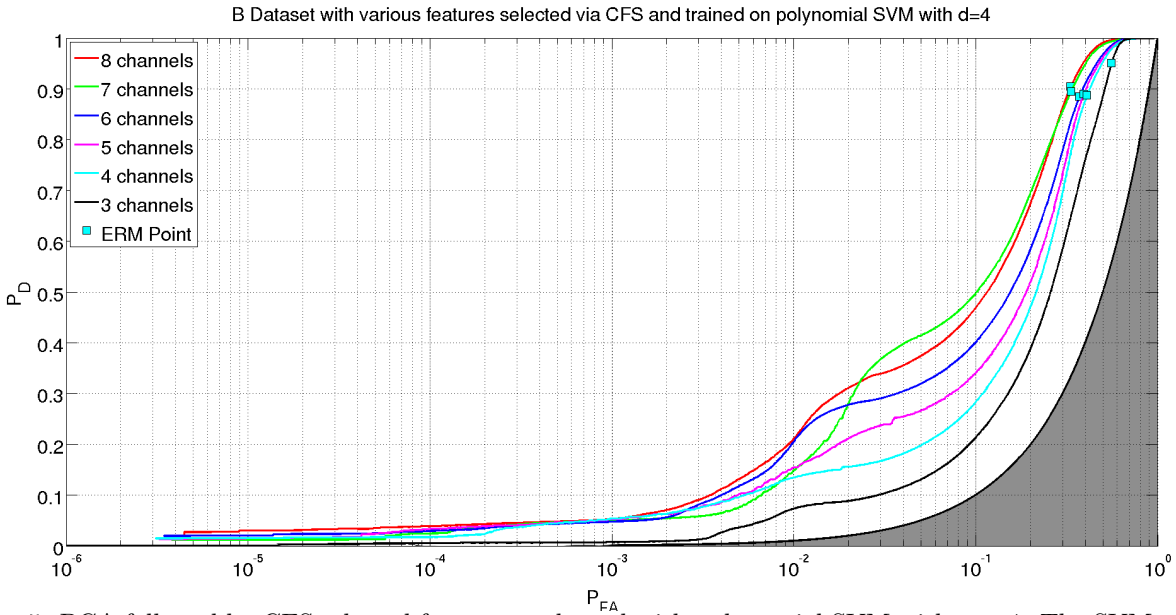


Figure 5: PCA followed by CFS selected features evaluated with polynomial SVM with $p = 4$. The SVM models were selected via ERM, note the ERM point annotated on each curve.

For the $p = 4$ case the parameter space was logarithmically gridded with $0.001 \leq a \leq 100,000$ and $0.01 \leq C \leq 500$. 375 parameter points were tested, requiring 375 training iterations. Total training time was ~ 25 hours on an Nvidia GTX 580 GPU with 3GB of RAM. PCA preconditioning was applied, and then CFS was used to select from 3 to 8 Mueller matrix elements after being transformed by PCA. The ROC curves for the best model for each number of CFS selected elements are shown in figure 5.

The empirical risk values are shown in the table below. In table 3, the ERM metric suggests that 4 features

Feature Selection Algorithm	Number of Channels	Kernel Type	Parameter Granularity	R_{emp}	$R(\alpha)$ Bound	$AUC_{0.1}$
PCA+CFS	8	Polynomial, $d = 4$	375 models	0.215	0.295	0.036
PCA+CFS	7	Polynomial, $d = 4$	375 models	0.219	0.281	0.037
PCA+CFS	6	Polynomial, $d = 4$	375 models	0.243	0.290	0.031
PCA+CFS	5	Polynomial, $d = 4$	375 models	0.250	0.286	0.025
PCA+CFS	4	Polynomial, $d = 4$	375 models	0.260	0.285	0.020
PCA+CFS	3	Polynomial, $d = 4$	375 models	0.304	0.321	0.013

Table 3: Performance of B datasets with CFS selection. ERM was used for the SVM model selection, but the $AUC_{0.1}$ metric was included for comparison purposes.

is nearly as good as 7 features (using the $R(\alpha)$ bound), but clearly the ROC curves show that the 7 feature curve is much better than 4 feature curve, and the $AUC_{0.1}$ metric ranks them correctly. Due to the ERM metric not performing as well as $AUC_{0.1}$ and the slow training of the polynomial SVM models, we decided to use $AUC_{0.1}$ in subsequent analysis.

4.3 Dataset C

The SVM training for C utilized the Gaussian kernel. The $AUC_{0.1}$ metric is used to evaluate performance. PCA preconditioning was applied, and then CFS was used to select from 3 to 8 Mueller matrix elements after being transformed by PCA and compared with the 16 (full Mueller matrix) ROC curve. The ROC curves for the best model for each number of CFS selected elements are shown in figure 6. The C ROC curve for 16 features (the

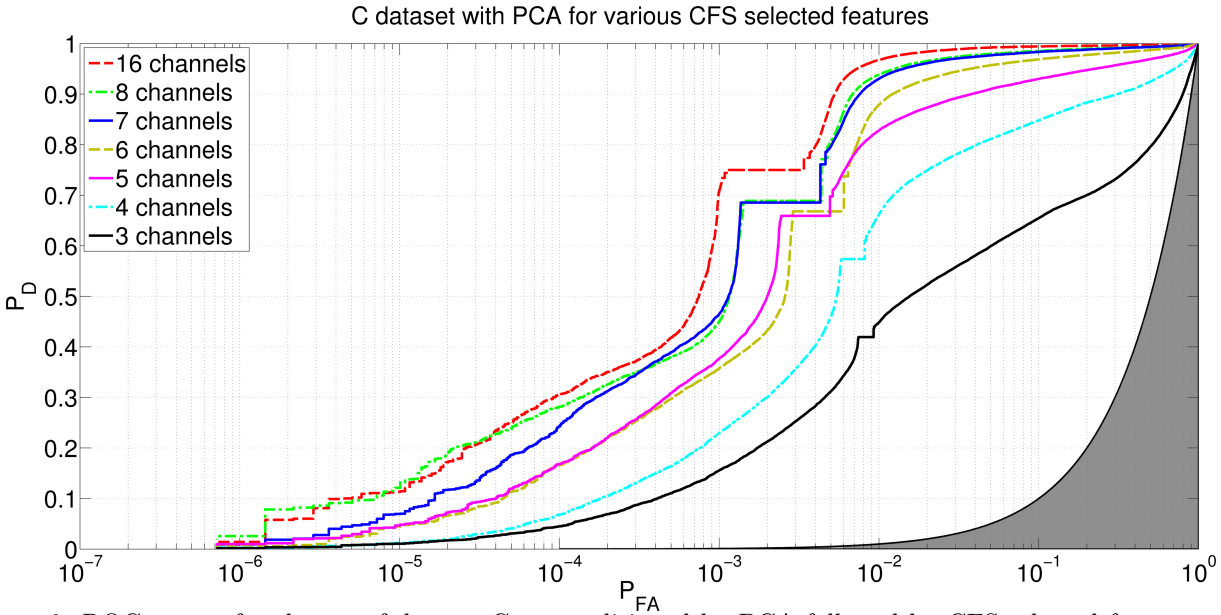


Figure 6: ROC curves for the set of dataset C preconditioned by PCA followed by CFS selected features. 3-8 and 16 features are shown. Note that $\sim 5\%$ of the C dataset was used for training, which is not adequate for training (see the last part of this section).

full Mueller matrix) is worse than those shown for the A dataset, but the TOIs were mixed with background for the C dataset. The ROC curves are quite good overall, considering that the materials were measured over BRDF parameters. We looked at the CFS selected ROC curves as well to determine how much information, in the form of Mueller elements, is actually needed to maintain a reasonable performance level.

Feature Selection Algorithm	Number of Channels	Kernel Type	Parameter Granularity	Number of Observations	$AUC_{0.1}$
PCA	16	Gaussian	1723 models	74,797	0.097
PCA+CFS	8	Gaussian	1627 models	74,597	0.095
PCA+CFS	7	Gaussian	1627 models	74,857	0.095
PCA+CFS	6	Gaussian	1723 models	75,258	0.093
PCA+CFS	5	Gaussian	1723 models	74,861	0.088
PCA+CFS	4	Gaussian	1723 models	74,899	0.078
PCA+CFS	3	Gaussian	1723 models	74,723	0.047

Table 4: Performance of C datasets with PCA transformation and CFS selection. The total number of observations in the data set is $1,495,570$.

Table 4 and figure 6 both show a jump in performance between 6 and 7 features, while the difference between 7 and 8 features is small, and still good compared with 16 features. This suggests that 7-8 features would be adequate, suggesting that a polarimeter only need to attempt to measure some subset of the total 16 elements in order to have good performance.

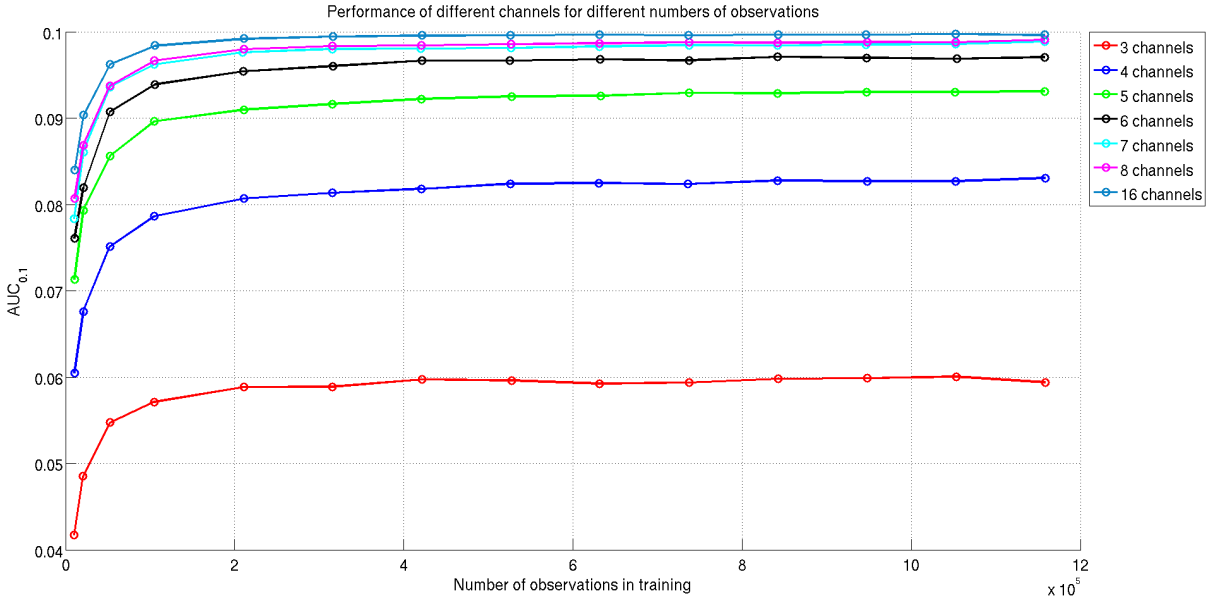


Figure 7: $AUC_{0.1}$ vs training set size. Note that the x-axis has a 10^5 factor. 3-8 and 16 features are shown.

An issue that arose during this analysis was the length of training the SVM models. Time to train is highly dependent on the size of the input dataset. This led to an experiment to assess how small the training set could become before performance ($AUC_{0.1}$) on the validation set began to degrade. This was done for the C dataset over all CFS features and is shown in figure 7. Performance remains about the same until around 200,000-300,000 Mueller matrices in the training set ($\sim 13-20\%$), then begins to drop off. This implies that adequate training will result from training set sizes of $\sim 13-20\%$ of the total C dataset. ROC curves for training with the full Mueller matrix are shown for different training set sizes in figure 8.

5. CONCLUSIONS

Non-linear classifiers have been shown here to result in good classification performance and ROC curves when applied to a large database of materials measured over BRDF parameters. SVM works quite well when using a Gaussian kernel, polynomial kernels result in subpar generalization and poor performance on the validation sets.

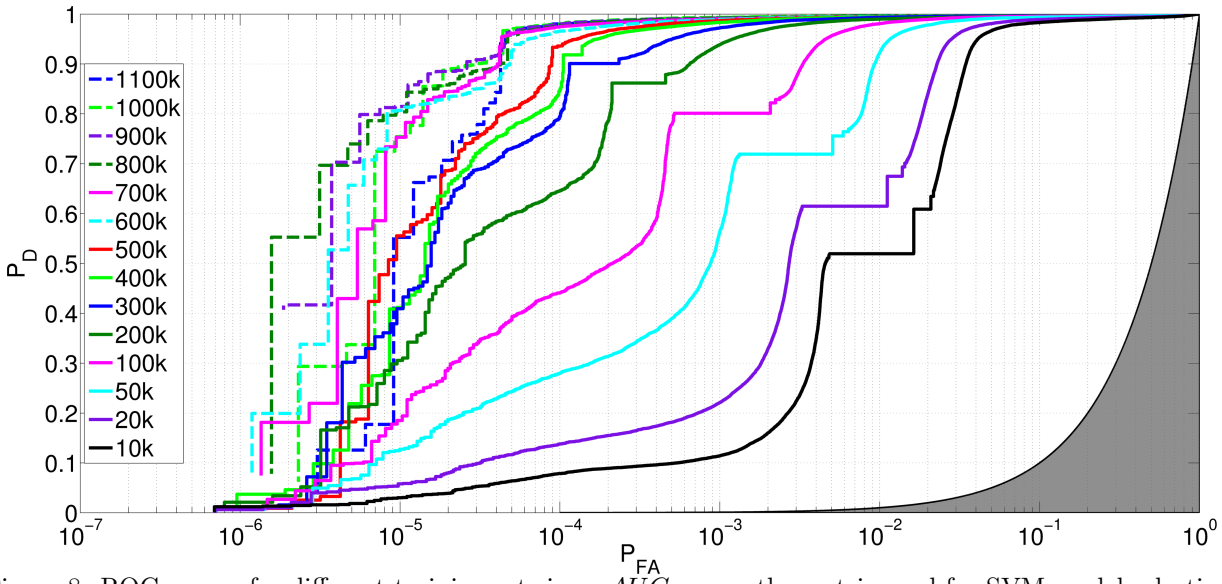


Figure 8: ROC curves for different training set sizes. $AUC_{0.1}$ was the metric used for SVM model selection.

In addition to classification on the full Mueller matrix, we have shown that for this dataset, performance degrades only slightly when using eight elements of the Mueller matrix (selected with CFS) instead of the full sixteen elements. This is important because instrument and hardware complexity can be reduced for field polarimeters, while losing little detection capability.

Future work includes applying different machine learning algorithms, like Riemannian manifold learning,[27] to the dataset, utilizing our algorithm in an imaging modality, and using machine learning to estimate material parameters instead of just classifying materials.

REFERENCES

[1] Jones, D. G., Goldstein, D. H., and Spaulding, J. C., "Reflective and polarimetric characteristics of urban materials," *Polarization: Measurement, Analysis, and Remote Sensing VII* **6240**(1), 62400A, SPIE (2006).

[2] Hoover, B. G. and Tyo, J. S., "Polarization components analysis for invariant discrimination," *Appl. Opt.* **46**, 8364–8373 (2007).

[3] Goudail, F., "Comparison of maximal achievable contrast in scalar, stokes, and mueller images," *Opt. Lett.* **35**, 2600–2602 (Aug 2010).

[4] Goudail, F. and Tyo, J. S., "When is polarimetric imaging preferable to intensity imaging for target detection?," *J. Opt. Soc. Am. A* **28**, 46–53 (Jan 2011).

[5] Anna, G., Goudail, F., and Dolfi, D., "Optimal discrimination of multiple regions with an active polarimetric imager," *Opt. Express* **19**, 25367–25378 (Dec 2011).

[6] Anna, G., Goudail, F., and Dolfi, D., "Polarimetric target detection in the presence of spatially fluctuating mueller matrices," *Opt. Lett.* **36**, 4590–4592 (Dec 2011).

[7] Weinreb, R. N., Zangwill, L., Berry, C. C., Bathija, R., and Sample, P. A., "Detection of glaucoma with scanning laser polarimetry," *Arch Ophthalmol* **116**(12), 1583–1589 (1998).

[8] Hall, M., "Correlation-based feature selection of discrete and numeric class machine learning," *Proceedings of the International Conference on Machine Learning* **17**, 359–366 (2000).

[9] Cloude, S. and Pottier, E., "An entropy based classification scheme for land applications of polarimetric sar," *Geoscience and Remote Sensing, IEEE Transactions on* **35**, 68 –78 (jan 1997).

[10] Ferro-Famil, L., Pottier, E., and Lee, J.-S., "Unsupervised classification of multifrequency and fully polarimetric sar images based on the h/a/alpha-wishart classifier," *Geoscience and Remote Sensing, IEEE Transactions on* **39**, 2332 –2342 (nov 2001).

[11] LEE, J. S., GRUNES, M. R., and KWOK, R., "Classification of multi-look polarimetric sar imagery based on complex wishart distribution," *International Journal of Remote Sensing* **15**(11), 2299–2311 (1994).

[12] Pierce, L., Ulaby, F., Sarabandi, K., and Dobson, M., "Knowledge-based classification of polarimetric sar images," *Geoscience and Remote Sensing, IEEE Transactions on* **32**, 1081 –1086 (sep 1994).

[13] Ulaby, F., Sarabandi, K., and Nashashibi, A., "Statistical properties of the mueller matrix of distributed targets," *Radar and Signal Processing, IEE Proceedings F* **139**, 136 –146 (apr 1992).

[14] van Zyl, J., "Unsupervised classification of scattering behavior using radar polarimetry data," *Geoscience and Remote Sensing, IEEE Transactions on* **27**, 36 –45 (jan 1989).

[15] Vaughn, I. J. and Hoover, B. G., "Noise reduction in a laser polarimeter based on discrete waveplate rotations," *Opt. Express* **16**, 2091–2108 (Feb 2008).

[16] Goldstein, D. H. and Chipman, R. A., "Error analysis of a mueller matrix polarimeter," *J. Opt. Soc. Am. A* **7**, 693–700 (Apr 1990).

[17] Smith, M. H., "Optimization of a dual-rotating-retarder mueller matrix polarimeter," *Appl. Opt.* **41**, 2488–2493 (May 2002).

[18] Twietmeyer, K. M. and Chipman, R. A., "Optimization of mueller matrix polarimeters in the presence of error sources," *Opt. Express* **16**, 11589–11603 (Jul 2008).

[19] Takakura, Y. and Ahmad, J. E., "Noise distribution of mueller matrices retrieved with active rotating polarimeters," *Appl. Opt.* **46**, 7354–7364 (Oct 2007).

[20] Cortes, C. and Vapnik, V., "Support vector networks," *Machine Learning* **20**, 273–297 (1995).

[21] Burges, C. J., "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery* **2**, 121–167 (1998).

[22] Smola, A. J. and Schölkopf, B., "A tutorial on support vector regression," *Statistics and Computing* **14**, 199–222 (2004). 10.1023/B:STCO.0000035301.49549.88.

[23] Lax, P. D., [*Functional Analysis*], Wiley-Interscience, New York, New York (2002).

[24] Stakgold, I., [*Boundary value problems of mathematical physics*], SIAM (2000).

[25] Barrett, H. H. and Myers, K. J., [*Foundations of Image Science*], Wiley-Interscience, Hoboken, New Jersey, United States (2004).

[26] Catanzaro, B., Sundaram, N., and Keutzer, K., "Fast support vector machine training and classification on graphics processors," in [*Proceedings of the 25th International Conference on Machine Learning*], *ICML '08*, 104–111, ACM, New York, NY, USA (2008).

[27] Lin, T. and Zha, H., "Riemannian manifold learning," *IEEE Trans. Pattern Anal. Mach. Intell.* **30**, 796–809 (2008).