

Classification of Biomedical Data Through Model-based Spatial Averaging *

Keith Marsolo
Srinivasan Parthasarathy
The Ohio State University
Department of Computer Science and Engineering
Contact: srini@cse.ohio-state.edu

Michael Twa
Mark A. Bullimore
The Ohio State University
College of Optometry

Abstract

Ensemble learning is frequently used to reduce classification error. The more popular techniques draw multiple samples from the training data and employ a voting procedure to aggregate the decisions of the classifiers constructed from those samples. In practice, such ensemble methods have been shown to work well and improve accuracy. Here we present a meta-learning strategy that combines the decisions of classifiers constructed from spatial models taken at multiple resolutions. By varying the resolution from coarse to fine-grained, we are able to partition the data on global features that describe a majority of the objects, as well as small, local features that are present in just a few problem cases. We test our technique on a biomedical dataset containing surface elevation values for diseased and non-diseased corneas. We transform these elevations into a series of coefficients using two different spatial transformations. Using these coefficients, we determine how well they distinguish between the two classes. We find our algorithm can increase the classification accuracy of a single decision tree up to 10% and can also be used in conjunction with traditional meta-learning techniques such as bagging to further improve performance. In an attempt to improve the execution time of the transformation algorithms, we have developed a distributed, grid-based implementation as well.

1 Introduction

One of the most promising applications of data mining and machine learning is their use on biomedical instrument data to assist clinicians in patient diagnosis. Any method of data analysis intended to support the clinical decision-making process should meet several criteria: it should capture clinically-relevant features, should be computationally feasible, and should provide easily interpretable results. Often there is a conflict between a system's ability to cap-

ture clinically-relevant features (i.e. provide high accuracy) and to provide easily interpretable results. A system can be highly tuned for a specific problem, but obtaining a simple rationale for a classification decision may be impossible. Support vector machines and neural networks are two examples of this type of "black-box" classification system. Decision trees are often preferred by clinicians because they are easily interpretable. To determine why a patient was classified in a particular manner, a user simply needs to trace through the tree. A single decision tree might not be accurate enough, however. Domain experts state that in order to address malpractice concerns and be trusted by their community, a classification system should provide high classification accuracy (at least 90%) on a consistent basis. Therefore, one would like a way to increase a classifier's accuracy while still maintaining some form of interpretability. One such strategy is to use *meta-learning*.

Meta-, or *ensemble-learning* techniques combine the output of several classifiers to improve results. Meta-learning is based on the theory that one can combine a number of classifiers with a larger error to create an overall classifier with a smaller error. The underlying intuition is that a group of experts usually provides a more accurate decision than any single expert, since their mistakes are likely to be uncorrelated. In this paper we look into the problem of patient diagnosis using decision trees and whether we can increase classification accuracy with meta-learning. We describe a new ensemble-learning strategy that uses multiple resolutions of the dataset in order to improve performance.

Given a dataset containing corneas labeled as diseased, normal and post-operative LASIK (the last two combined into a single "non-diseased" class), we use two different spatial transformations to model the corneal surface. Both methods have been studied by the medical community and have been found to provide an anatomically-faithful model. In addition, there is a relationship between the lower polynomial modes of the series and aberrations of the cornea. We take the models produced by these transformations and use them as input to a decision tree classifier. We combine the classifiers with our meta-learning technique to evaluate its

*NIH-NEI T32-EY13359, EY16225 and American Optometric Foundation William Ezell Fellowship, Ocular Sciences Inc. (MT); NSF CAREER Grant IIS-0347662(SP); NIH-NEI R01-EY12952 (MB).

effect on performance. *We find that this method provides results superior to those returned by a single decision tree, and when used in conjunction with traditional meta-learning strategies, can further increase classification accuracy.*

Finally, we have implemented high-performance versions of our transformation algorithms that take advantage of the benefits of distributed computing. Designed to utilize a workstation’s spare computing cycles, we can achieve high-performance results without a dedicated computing infrastructure, making our solution ideal for a health center environment without access to a supercomputer. We implement the high-performance version of our algorithms using a framework called NetSolve [1], achieving speedups of 13x-50x. *To the best of our knowledge, our optimizations result in the fastest implementation of this particular application, both commercially and publicly-available.*

2 Traditional Meta-Learning Approaches

Two of the more popular meta-learning techniques are boosting [9] and bagging [3]. As is the case with many meta-learning strategies, boosting and bagging reach a decision through the aggregation of multiple hypotheses. Both methods take a user-specified classification algorithm and construct a number of different classification models (also user-specified). The results of these models are combined to reach a final decision. When constructing the classifiers, a different sample of the data is selected (with replacement) at each iteration.

The difference between the two techniques is that bagging draws an independent sample from the data at every iteration, whereas boosting draws a weighted sample, with misclassified objects carrying a higher weight during the training process. Since a misclassification represents an object that was not properly captured by the underlying model, more emphasis is placed on those objects that are “problematic.” In addition to drawing a sample that is biased toward misclassified objects, when the final classifier combines the results of the underlying classifiers, boosting puts more weight on the classifiers built from the samples containing misclassified objects, as they are believed to be more accurate. Bagging, on the other hand, weighs the contribution of all underlying classifiers equally. Since both techniques draw random samples from the data at each iteration, they are most effective when paired with “unstable” classifiers, or classifiers that are sensitive to minor perturbations in the training data.

3 Spatial Representations

The dataset used in our tests consists of patient evaluations experimentally obtained from a corneal topographer. The absolute number of data points is a function of the topographer’s sampling resolution, which should be high enough to capture any surface aberrations that might be present. It is not possible, however, to use this raw data as a feature vec-

tor for classification. There is no way to guarantee that point X of one record refers to the same surface location as point X of another. Therefore, in order to make a true “apples to apples” comparison, we must first transform the data.

3.1 Zernike Polynomials

Zernike polynomials are a family of circular polynomial functions that are *orthogonal* in x and y over a normalized unit circle and consist of three elements [12]. The first element is a normalization coefficient. The second element is a radial polynomial component and the third, a sinusoidal angular component. The general form for Zernike polynomials is given by:

$$Z_n^{\pm m}(\rho, \theta) = \begin{cases} \sqrt{2(n+1)}R_n^m(\rho) \cos(m\theta) & \text{for } m > 0 \\ \sqrt{2(n+1)}R_n^m(\rho) \sin(|m|\theta) & \text{for } m < 0 \\ \sqrt{(n+1)}R_n^m(\rho) & \text{for } m = 0 \end{cases} \quad (1)$$

where n is the radial polynomial order and m represents azimuthal frequency. The normalization coefficient is given by the square root term preceding the radial and azimuthal components. The radial component of the Zernike polynomial is defined as:

$$R_n^m(\rho) = \sum_{s=0}^{(n-|m|)/2} \frac{(-1)^s (n-s)!}{s! \binom{n+|m|}{2} (-s)! \binom{n-|m|}{2} (-s)!} \rho^{n-2s} \quad (2)$$

Polynomials that result from fitting our raw data with these functions are a collection of orthogonal circular geometric modes. When combined, these modes form a surface that faithfully models the shape of the cornea. The coefficients of each mode are proportional to their contribution to the overall topography of the original data. As a result, we can effectively reduce the dimensionality of the data to a subset of polynomial coefficients that describe spatial features, or a proportion of specific geometric modes present in the original data.

3.2 Pseudo-Zernike Polynomials

Pseudo-Zernike polynomials, also called Bhatia-Wolf polynomials, are another family of orthogonal polynomial functions [2, 11]. Like Zernike polynomials, these functions are a series of circular modes orthogonal over x and y for a normalized unit circle. Pseudo-Zernike polynomials are additionally orthogonal in the radial dimension ($\rho = \sqrt{x^2 + y^2}$) over a unit circle. The general form for pseudo-Zernike polynomials is given by:

$$PZ_n^{\pm m}(\rho, \theta) = \begin{cases} \sqrt{2(n+1)}R_n^m(\rho) \cos(m\theta) & \text{for } m > 0 \\ \sqrt{2(n+1)}R_n^m(\rho) \sin(|m|\theta) & \text{for } m < 0 \\ \sqrt{(n+1)}R_n^m(\rho) & \text{for } m = 0 \end{cases} \quad (3)$$

Transform	4	5	6	7	8	9	10
Zernike	15	21	28	36	45	55	66
Pseudo-Zernike	25	36	49	64	81	100	121

Table 1. Total number of coefficients for each transformation as a function of polynomial order.

Similarly, the radial portion of the pseudo-Zernike polynomial is defined as:

$$R_n^m(\rho) = \sum_{s=0}^{n-|m|} \frac{(-1)^s (2n+1-s)!}{s!(n-|m|-s)!(n+|m|+1-s)!} \rho^{n-s} \quad (4)$$

It has been shown that for the same polynomial order, pseudo-Zernike polynomials provide a richer, more accurate surface model than Zernike polynomials [6]. To date, their performance in classification has not been examined. One disadvantage of using pseudo-Zernike polynomials is their greater computational cost, due to the increased number of terms per order. With both transformation methods, there is a relationship between the lower polynomial modes and aberrations of the cornea.

When discussing our data transformations, we often refer to Zernike or pseudo-Zernike polynomials of a certain order. Each of these polynomials are comprised of a varying number of coefficients. We provide the number of coefficients for each order and transformation that we tested in Table 1. Since the series are orthogonal, all of the lower order coefficients are contained within the higher order transformations. For example, a 5th order Zernike polynomial consists of 21 coefficients, but the first 15 are also contained in the 4th order transformation.

4 Implementation Details

Our data transformation is based on methods described in detail by Schwiegerling et al. and Iskander et al. [10, 5]. In summary, the data is modeled using either polynomial transformation method over a user-selected circular region of variable-diameter centered on the axis of measurement. Modeling proceeds in an iterative fashion, computing a point-by-point representation of the original data at each radial and angular location up to the user-specified limit of polynomial complexity. The polynomial coefficients of the model that will later be used to represent the proportional magnitude of specific geometric features are computed by performing a least-squares fit of the model to the original data, using standard matrix inversion [10].

Figure 1(a) contains a pseudo-code description of the algorithm we use to compute the Zernike and pseudo-Zernike

polynomial transformations for each patient record. The algorithm input consists of the user-defined variables *numCoefs*, the number of coefficient terms to compute (based on the polynomial order, see Table 1), and *maxRadius*, the largest radius to examine. The original version of our algorithm was implemented using Matlab¹.

The first steps in the pseudo-code (Steps 1 and 2, Figure 1(a)) involve reading the radius and elevation values from the patient record and scaling them from *mm* to μm . Once this step is complete, we are left with arrays representing the radius and elevation values. Each concentric ring contains 256 points, so we know the angle for every radius value (determined by its index in the data file) and can compute an array of θ values that correspond to the angle of each data point.

The next step (Step 3, Figure 1(a)) is to determine the maximum number of data points that will be modeled. This is done by finding the first data value with a radius that exceeds the user-specified parameter *maxRadius*. Since this point can occur anywhere along a discrete ring of data points, we perform an integer division on that value and then multiply by 256 to ensure that we include a complete ring. We then consider all the data points in the patient record that come before this value, which we denote as *numDP*. All the experiments discussed here were run with a radius value of 3.5 *mm*.

Once we have determined the number of data points to examine, we perform a Zernike and pseudo-Zernike transformation (Figures 1(c) and 1(b), respectively) on each point. In Figures 1(b) and 1(c), *n* is the radial polynomial order and *m* represents azimuthal frequency. The values of *m* and *n* are determined based on the number of coefficients, *j*. The transformation process, shown in Step 5 of Figure 1(a) results in a matrix of (*numDP* × *numCoefs*) values. After computing the given transformation on each data point, we compute a least-squares estimation of our computed surface against the surface representing the height values from the original patient record. Computing the least-squares fit, shown in Step 6, results in a coefficient vector of length $|numCoefs|$. Finally, we compute the residual model error as a function of the polynomial order by performing a root-mean-square (RMS) error calculation between the actual surface and the model (Steps 7-9).

5 High-Performance Implementations

Running on a single PC, it took over a week to compute both spatial transformations on our dataset over all the polynomial orders tested. Thus, for our method to serve as a viable system of model-based classification, we need a faster, more efficient transformation algorithm. To that effect, we have implemented optimized, distributed versions of our code in order to decrease the computation time.

¹<http://www.mathworks.com/products/matlab/>

```

Compute_Coefficients(numCoefs,maxRadius)
1. Read in radius ( $\rho$ ) and elevation ( $z$ ) values from patient record.
2. Convert radius and elevation values from mm to microns.
3. Find index of first radius value that exceeds maxRadius.
4.  $numDP = index / 256 * 256$ 
5. for  $i = 1$  to numCoefs do
   for  $j = 1$  to numDP do
      $\rho = Radius[j]$ 
      $\Theta = Angle[j]$ 
      $M[j, i] = Coef(i, \rho, \theta)$  (PZ_Coef or Z_Coef)
   endfor
endfor
6.  $A = (M^T * M)^{-1} * M * Z$ 
   Z is a matrix containing the first numDP elevation values
   from the patient record.
7. ZT = floating-point vector of size |numDP|.
8. for  $i = 1$  to numCoefs do
   ZMatrix = column  $i$  of M
   ZMatrix = ZMatrix * A[i]
   ZT = ZT + ZMatrix
endfor
9. Compute RMS value of ZT.
    $RMS = \|Z - ZT\|_2 / \sqrt{numDP}$ 
10. Output results.

```

(a)

```

PZ_Coef( $i, \rho, \theta$ )
1. for  $s = 0$  to  $(n - |m|)$ 
    $Sum = Sum + \frac{(-1)^s (2n+1-s)!}{s!(n-m-s)!(n+m+1-s)!} \rho^{n-s}$ 
endfor
2. if  $m = 0$  then
    $z = \sqrt{n+1} * Sum$ 
else if  $m < 0$  then
    $z = \sqrt{2(n+1)} * Sum * \sin(m\theta)$ 
else
    $z = \sqrt{2(n+1)} * Sum * \cos(m\theta)$ 
3. return  $z$ 

```

(b)

```

Z_Coef( $i, \rho, \theta$ )
1. for  $s = 0$  to  $\frac{n-|m|}{2}$ 
    $Sum = Sum + \frac{(-1)^s (n-s)!}{s!(\frac{n+|m|}{2}-s)!(\frac{n-|m|}{2}-s)!} \rho^{n-2s}$ 
endfor
2. if  $m = 0$  then
    $z = \sqrt{n+1} * Sum$ 
else if  $m < 0$  then
    $z = \sqrt{2(n+1)} * Sum * \sin(|m|\theta)$ 
else
    $z = \sqrt{2(n+1)} * Sum * \cos(m\theta)$ 
3. return  $z$ 

```

(c)

Figure 1. Pseudo-code descriptions of (a) overall algorithm, (b) pseudo-Zernike transformation and (c) Zernike transformation

A profiling analysis of our Matlab algorithm revealed that almost all of the computation time was spent running through the **for** loops listed in step 5 of Figure 1(a), with the total time increasing along with the number of coefficients computed. For instance, step 5 took about 20 seconds for a 4th order Zernike polynomial, but 350 seconds on a 10th order pseudo-Zernike. The rest of the algorithm, on the other hand, usually took less than a second. We provide the time needed to execute step 5 for each transformation over each order in Figure 2(a). In the same figure, we show the computation time needed to finish step 5 in a native C++ implementation. The differences are startling, with the C++ version rarely taking more than two seconds and generally finishing in less than one.

We did not wish to abandon Matlab, as it has many benefits, such as ease of implementation and the ability to quickly test program changes. Also, for certain matrix and vector operations (i.e. the operations used to compute the least squares fit of our surface as well as RMS error), we found Matlab to be far superior to a native C++ implementation. We provide the running times for the C++ and Matlab versions of steps 6-9 in Figure 2(b). As shown in the figure, Matlab required 0.4 seconds on a 10th order Zernike polynomial and 1.2 for a 10th order pseudo-Zernike. On the same transformations, the C++ implementation took 8.4 and 30.1 seconds, respectively. The matrix and vector operations needed in steps 6-9

are not natively available in C++, so we used the BOOST² matrix and vector libraries as well as some of the matrix operations from the Automatically Tuned Linear Algebra Software (ATLAS)³ library.

With the above results in mind, we set out to design a version of the transformation algorithm that provided C++-like performance in step 5, yet still allowed us to harness the benefits of Matlab for steps 6-9. To that effect, we created an implementation of our algorithm that was integrated with the NetSolve framework. Here, functions written in C or FORTRAN are installed on a NetSolve computation server. A user makes calls to an agent that forwards the requests to the appropriate server. The results are returned upon completion.

We test the high-performance implementations with the following setup: Matlab was installed on a PC with an Athlon XP 1800 processor and 512 MB RAM running Windows XP. NetSolve was installed on a different PC, also with an Athlon XP 1800 processor and 512 MB RAM, but running Debian Linux with a 2.4.x kernel. Both computers were on the same network connected through a 100 Mbps Ethernet switch. NetSolve was set to run as a service, where it would wait in the background for incoming requests. A C version of step 5 was created and installed on the NetSolve server. Step 5 of the original Matlab code was replaced to

²<http://www.boost.org>

³<http://math-atlas.sourceforge.net>

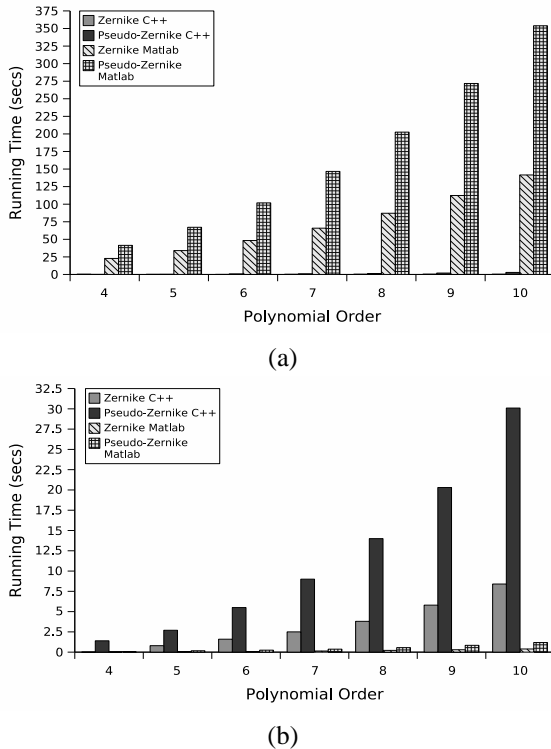


Figure 2. (a) Average execution time (in seconds) for step 5 of Fig. 1(a). (b) Average execution time (in seconds) for steps 6-9 of Fig. 1(a). Results are given for C++ and Matlab implementations over each polynomial order.

include a call to the NetSolve agent.

Adding the times in Fig. 2 (a) and (b) provides the running time for both the unoptimized transformation algorithm and the distributed NetSolve version. Using NetSolve drastically reduces the computation time, providing a minimum speedup of at least 13x. To compute a 10th order pseudo-Zernike polynomial, the longest-running transformation, the difference in computation time is staggering. The unoptimized version takes almost six minutes to complete. The version integrated with NetSolve takes roughly six *seconds*, a speedup of more than 50x. Since the speed-up factor tends to increase as the polynomial order of the transformation increases, the computational benefits would likely be even greater for transformations involving polynomial orders larger than ten.

6 Spatial Model-Averaging Meta-Learning

We now present our meta-classification strategy, which is based on the idea of averaging the results from classifiers built on multiple spatial representations to reach a decision. We refer to this strategy as Spatial Averaging (SA). Unlike boosting and bagging, which use multiple classifiers con-

structed from a single data model using random sub-samples of the data to improve classification accuracy, we use the results of classifiers built using multiple spatial models. The strategy was designed to smooth out perturbations due to *noise effects*. Intuitively, the lower order spatial transformations result in a model that is less detailed than those generated by the higher order transformations. However, the higher order transformations also contain noise that is not present in the models derived from the lower order transformations. Similarly, as the pseudo-Zernike transformation yields a richer representation (for the same order) than a Zernike polynomial, *by combining the results of classifiers generated from the two representations, we may further minimize noise effects*.

The Spatial Averaging algorithm is as follows: First, each individual record is converted into an input appropriate for the underlying model classifier. This step is repeated on each record for each classifier that is to be aggregated by the SA-classifier. Second, a vote for a particular classification label is determined for each record by running the converted input through every model classifier. Finally, the SA-classifier makes a decision by taking the majority label of the votes returned by the model classifiers.

The motivation behind Spatial Averaging is similar to that of bagging. Breiman notes that bagging works by creating an aggregate predictor based on a sequence of underlying predictors formed from several replicates of the training data, or in cases where those replicates do not exist, by taking a series of random samples from the original training set [3]. By coupling the different samples with an unstable classifier, the goal is to cause perturbations in the predictor that will lead to improvements in the overall classification accuracy. The same principle applies to Spatial Averaging. By using an increasingly detailed data transformation over the *same* sample, we add additional features to our classification input that will (potentially) help distinguish problem cases. To use Breiman’s terminology, we can consider each polynomial order as a replicate of the original training data. The 4th order polynomials are one replicate, the 5th order a second replicate, and so on. Therefore, instead of having to take random samples to obtain our replicate training data, we can simply apply different transformations to the same sample.

In his review on bagging and its ability to increase classification accuracy, Domingos argues that the improvement is due to a shift in complexity from an overly simplistic decision tree model to one that is more complex [4]; in effect, the supposedly beneficial *simplicity bias* that is a function of many decision tree algorithms can actually degrade classification performance. A simple decision tree may be more interpretable, but limiting or favoring a small number of leaves can cause a “smoothing over” of the learner’s model. A tree with a large number of leaves might be able to more accurately partition this space, but a more simplistic and therefore

coarser-grained model is preferred. While bagging achieves this increase in complexity through resampling, we achieve a similar increase by modeling the data at varying resolutions.

7 Dataset

The dataset for these experiments consists of the examination data of 254 eyes obtained from a clinical corneal topography system. The data were examined by a clinician and divided into three groups based on corneal shape. The divisions are given below, with the number of patients in each group listed in parentheses:

1. Normal ($n = 119$)
2. Post-operative myopic LASIK ($n = 36$)
3. Keratoconus ($n = 99$)

Relative to the corneal shape of normal eyes, post-operative myopic LASIK corneas generally have a flatter center curvature with an annulus of greater curvature in the periphery. Corneas that are diagnosed as having keratoconus are also more distorted than normal corneas. Unlike the LASIK corneas, they generally have localized regions of steeper curvature at the site of tissue degeneration, but often have normal or flatter than normal curvature elsewhere. While it is possible to treat this as a multi-class decision problem, we elect to combine the Normal and post-LASIK corneas into the same class. The reason for this decision is that we want to differentiate diseased corneas from those that are non-diseased.

One of the purported benefits of pseudo-Zernike polynomials is that they provide a richer representation. Thus, for a given polynomial order, they should be able to produce a surface model with a smaller root mean square (RMS) error. Groups have examined the use of pseudo-Zernike polynomials to model the corneal surface [6]. To date, however, no one has reported on their utility as a method for data transformation before classification. There are tradeoffs between classification accuracy and the ability of the polynomials to model the shape of the cornea. Since there is no consensus in the literature as to which order is “best,” we test the transformations on a number of orders. *We found that for a given polynomial order, the models produced by the pseudo-Zernike transformation consistently resulted in a smaller RMS error than the models produced by the Zernike transformation.* As an example, for a 7th order transformation, the average RMS error for the Keratoconic patients was $0.42 \mu\text{m}$ with a Zernike transformation and $0.30 \mu\text{m}$ with the pseudo-Zernike. In addition, as the order increased, the corresponding RMS error decreased. This was true for both transformations. Whether a higher fidelity surface model translates into better classification accuracy is unknown.

8 Experiments and Results

In this section we detail the performance of our meta-learning algorithm. Unless otherwise noted, all results are listed as the percentage of correct classification of the testing data.

Like many meta-learning algorithms, our Spatial Averaging technique can be applied to a number of underlying classification strategies. In these experiments, we elected to use C4.5 decision trees [8]. This choice was made for several reasons. Decision trees are fast, easy to construct, and highly interpretable, which is a plus when trying to explain the criteria for a classification decision to a member of the medical community.

8.1 Spatial Averaging vs. C4.5

The first set of experiments involve testing the performance of our Spatial Averaging technique against a standard decision-tree-based classifier. The standard decision tree is intended to serve as a baseline. We want to examine how well our SA strategy performs as a meta-learning technique.

In these experiments, we constructed a Spatial Averaging classifier using the algorithm detailed in Section 6. The Zernike and pseudo-Zernike-based classifiers were created by constructing a decision tree for each transformation over each polynomial order tested (we stopped at 10 as classification accuracy plateaued at that point). Each patient record was converted into a vector of coefficients taken from the Zernike or pseudo-Zernike representation of that order. Each vector was then run through its corresponding decision tree. The votes or labels of each tree were tallied and the majority vote was taken as the decision of the SA-classifier.

The results for each technique were averaged over ten separate runs. The dataset was divided into a 2/3 - 1/3 training/testing split. For a given run, the same testing split was used and the various training sets were generated by selecting a random sample (with replacement) from the training data. For each underlying tree of the SA-classifier (i.e. 4th-10th order trees), the same training data was used when generating each tree, though the data was transformed using the appropriate method. A different testing split was generated at random for each of the ten runs.

As shown in Figure 3, the Spatial Averaging technique provides much higher accuracy than standard C4.5. The accuracy of a Zernike-based Spatial Averaging classifier provides an improvement ranging from 3-5% over a traditional C4.5 decision tree, and a pseudo-Zernike-based SA-classifier provides up to a 7% improvement. We can further improve accuracy by creating a Spatial Averaging classifier that combines the Zernike and pseudo-Zernike trees before making a decision. This classifier would take a vote of the 14 individual trees and make a decision based on the majority label (a tie is counted as a misclassification). Such a “combined” classifier improves classification accuracy to

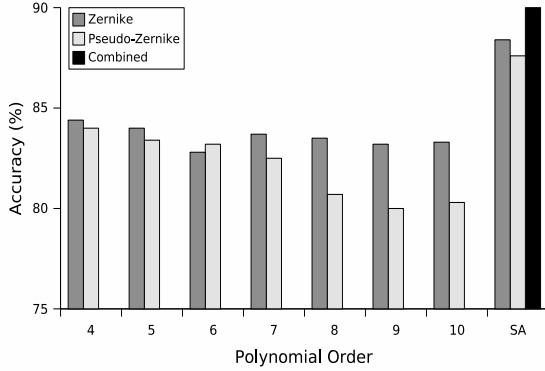


Figure 3. Accuracy for Spatial Averaging (SA) compared with C4.5. Values are given for each order and transformation. Also shown are the results for the combined SA classifier.

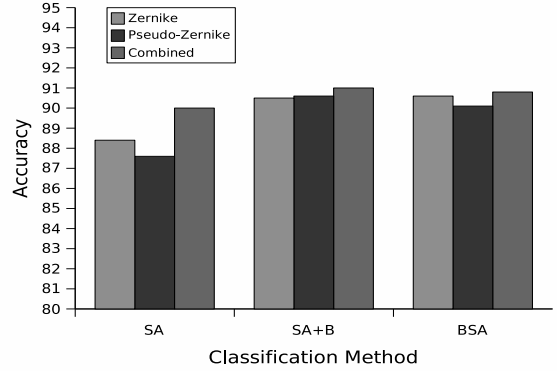


Figure 4. Accuracy for SA-classification combined with meta-learning techniques. SA corresponds to Spatial Averaging, SA + B to SA + Bagging and BSA refers to Bagged SA.

90%. This is a full 10% improvement over the lowest individual C4.5 classifier and close to 6% higher than the best individual tree. It should be noted that despite the pseudo-Zernike transformation providing a surface with a smaller error, this increase in model fidelity does not translate into an increase in classification accuracy. In fact, for a given polynomial order, the accuracy of a pseudo-Zernike decision tree is roughly the same or even less than the corresponding Zernike classifier.

8.2 Spatial Averaging and Meta-Learning

While we would expect our Spatial Averaging algorithm to perform well compared to traditional C4.5, we would also like to determine how it compares to other meta-learning techniques. First, we want to directly compare how Spatial Averaging fares against traditional meta-learning strategies such as boosting and bagging. To that effect, we test each method on both transformations using the 4th-10th order datasets. We find the performance of the standard meta-learning techniques to be roughly equivalent on each order, averaging about 85%. The highest accuracy for each of these methods was around 88% with a 4th order Zernike transformation. This compares with an accuracy of 90% for the combined SA-classifier, a 2% gain. We test each meta-learning algorithm over a varying number of iterations (2-20). Since the combined SA-classifier averages the results of 14 decision trees, we want to include at least that many iterations in our tests.

Second, we would also like to determine whether it is possible to use existing meta-learning techniques in conjunction with Spatial Averaging in order to improve classification accuracy over a traditional SA-classifier or a standard C4.5-based meta-learner. We elected to implement two different “bagged” versions of our SA-classifier. With the first version, which we denote as *Spatial Averaging + Bagging*

(SA + B), classification occurs as a two step process. In the first step, several different decision trees for each polynomial order are created and a final decision is made for those trees. The next step involves aggregating the decisions of each order’s ensemble and taking the majority label as the final decision. We elect to use 11 trees for each polynomial order. Each of the 11 trees is generated using a different random sample of the training data, but the same random sample is used when generating tree X for each order. We create a tree for each individual transformation (Zernike and pseudo-Zernike) as well as for a “combined” transformation (as above).

The second implementation involves the creation of several individual SA-classifiers (using the procedure described in Section 6) and taking the majority vote of those trees as the final classification decision. We call this implementation *Bagged Spatial Averaging* (BSA). To test these implementations, we again divide the dataset into 2/3 - 1/3 training/testing split and average the results over ten different runs. The results are presented in Figure 4. For an individual transformation, SA + B provides a 2-3% increase over a standard SA-classifier. The improvement falls to 1% for the combined classifier, but there is still an improvement in accuracy. Similar results are seen with the BSA-based classifier.

Meta-learning techniques such as boosting and bagging increase accuracy by resampling the training data. By drawing enough samples, the intent is to increase the predictive or descriptive ability of the overall classification model. If one viewed each object as a row in a matrix and each attribute in that object’s feature vector as a column, boosting and bagging sample from the rows. This is why it is not practical to combine boosting with bagging or vice versa, as both algorithms use similar designs to reduce error. Spatial Averaging

on the other hand, targets the column space. Therefore, it can be coupled with techniques such as bagging, providing a greater benefit and a larger increase in predictive ability than any single ensemble-learner.

9 Discussion

Here we present our algorithm for meta-learning classification through model-based spatial averaging. Unlike traditional meta-learning techniques, which work to increase accuracy by continually sampling the training data, our method looks to achieve the same result by modeling the data at varying resolutions. As a result of this difference, our technique can be coupled with traditional meta-learning methods in order to further improve performance. Empirical results have shown that our standard algorithm can increase accuracy more than 10% over a single decision tree, and when coupled with other meta-learners, can provide an additional increase of almost 5%.

We also provide the first in-depth study on the use of pseudo-Zernike polynomials in classification. Our general finding was that unlike modeling results reported in the literature [6], *there was no special benefit to using pseudo-Zernike polynomials over Zernike polynomials in classification*. Classification accuracy using pseudo-Zernike polynomials was roughly the same or slightly worse than a Zernike polynomial of the same order. Part of the allure of pseudo-Zernike polynomials is that they are more robust to noise by including more terms for a given order. If these terms were modeling important, distinguishing features, classification accuracy would improve as the polynomial order was increased. Using a single C4.5 decision tree, the highest accuracy was achieved when using a 4th order Zernike transformation as input. The use of pseudo-Zernike polynomials does help significantly improve the accuracy of the Spatial Averaging classifier, but if one has to use a *single* decision tree, a low order Zernike polynomial will suffice.

While accuracy is the most important factor in choosing a classification strategy, another attribute that should not be ignored is the overall interpretability of the final results. Part of the rationale behind using Zernike polynomials as a transformation method over other alternatives is that there is a direct correlation between the geometric modes of the polynomials and the surface features of the cornea. We have shown our decision trees to practicing clinicians and preliminary validation from them indicates there is an anatomic correspondence between the features of these trees and corneal aberrations of patients with keratoconus. In addition, we have created a visualization technique that can be used to illustrate the criteria used in decision-tree classification [7]. While our visualization currently applies to single decision trees, we are working to implement a version that can handle aggregate trees as well.

We have also provided a high-performance implementa-

tion of our transformation algorithm that results in a drastic reduction in the overall running time and also provides a foundation for a distributed implementation of our classification system within a hospital or health center.

In the future we plan to conduct additional experiments with our Spatial Averaging algorithm. We plan to compare it to traditional meta-learners to determine whether there are certain “problem cases” that we can classify correctly on a more consistent bases. We also believe this technique may be useful for any problem where the features used in classification are derived with a model-based approach. To that effect, we would like to expand the scope of our experiments to other datasets and transformation methods, such as wavelets and FFTs, to determine the type of improvements we can gain there as well.

References

- [1] D. Arnold, S. Agrawal, et al. Users' guide to netsolve v1.4.1. ICD TR ICL-UT-02-05, Uni. of TN, Knoxville, TN, June 2002.
- [2] A. B. Bhatia and E. Wolf. On the circle polynomials of zernike and related orthogonal sets. In *Proc. Cambridge Philos. Soc.*, volume 50, pages 40–53, 1954.
- [3] L. Breiman. Bagging predictors. *Machine Learning*, 26(2):123–140, 1996.
- [4] P Domingos. Why does bagging work? a bayesian account and its implications. In *Third International Conference on Knowledge Discovery and Data Mining*, pages 155–158, Newport Beach, CA, 1997. AAAI Press.
- [5] DR Iskander, MJ Collins, and B Davis. Optimal modeling of corneal surfaces with zernike polynomials. *IEEE Trans Biomed Eng*, 48(1):87–95., 2001.
- [6] DR Iskander, MR Morelande, and B Collins, MJ Davis. Modeling of corneal surfaces with radial polynomials. *IEEE Trans Biomed Eng*, 49(4):320–328, 2002.
- [7] K. Marsolo, M.D. Twa, S. Parthasarathy, and M. Bulimore. A model-based approach to visualizing classification decisions for patient diagnosis. In *10th Conf. on AI in Medicine (AIME)*, Aberdeen, Scotland, 2005.
- [8] J.R. Quinlan. *C4.5 : programs for machine learning*. Morgan Kaufmann Publishers, San Mateo, Calif., 1993.
- [9] R.E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.
- [10] J. Schwiegerling, J. E. Greinvenkamp, and J. M. Miller. Representation of videokeratoscopic height data with zernike polynomials. *J. Opt. Soc. Am. A*, 12(10):2105–2113, 1995.
- [11] C-H Teh and RT Chin. On image analysis by the methods of moments. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10(4):496–513, 1988.
- [12] LN Thibos, RA Applegate, and R Schwiegerling, JT Webb. Standards for reporting the optical aberrations of eyes. In *TOPS*, Santa Fe, NM, 1999. OSA.