Classification study of three plant species

Kehinde Oyemakinwa, Shafiul Alam, Aleksi Leinonen

Laboratory Practice Report
March 2018
Department of Physics and Mathematics
University of Eastern Finland

K. Oyemakinwa, S. Alam, A. Leinonen *Classification study of three plant species*, 12 pages
University of Eastern Finland
Master's Degree Programme in Photonics

Supervisors Ph.D. Hannu Laamanen

Abstract

The possibility of using machine learning algorithms to analyze spectral data of three different plants in infrared region was analyzed in this study. Machine learning code was developed utilizing average learning subspace method (ALSM) and principal component analysis (PCA). This code was then used on spectral cubes of birch, spruce and pine. These spectral cubes were aquired using a spectral line camera working in infrared region of $1000 - 2500$ *nm.*

The code identified all test samples correctly after four iterations in learning phase. In testing phase two samples were used: one with all plants separated in one image and another with all samples mixed together. Most pixels of these samples were identified correctly but some parts of spruce and pine were misclassified. Intrestingly birch was always classified correctly. These results were propably caused by the differences in spectral behaviour of samples.

Contents

Chapter I

Introduction

Modern spectroscopic measuring methods are widely used in biological and chemical studies. These methods provide a luxury of information, which can become difficult to analyze. Machine learning has become efficient in analyzing this spectroscopic data, thanks to increased computing power of modern computers and more efficient machine learning methods. Especially PCA technique has proven to increase the efficiency of machine learning in classification of high-dimensional data. [1–3]

In this study three plants are chosen and measured using a hyperspectral camera in infrared region. Plants will be classified by using ALSM with PCA. This machine learning technique will be taught to identify the chosen plants. After learning phase, the code will be used to test how it can identify a mixture of used plants.

Chapter 2 provides the basic information regarding the spectral information of plants in infrared region and the basic theory behind PCA and ALSM techniques. In Chapter 3 the measurements and development of machine learning code will be presented. Chapter 4 will focus on the testing of the code and discussion of the results. Finally, in Chapter 5 the conclusions will be given.

Chapter II

Theory

In this Chapter the relevance of infrared radiation to spectral properties of plants will be reviewed. Also, the theory of PCA and ALSM will be introduced.

2.1 Spectral properties of plants

Plant is known as a solar energy converter as it depends on the sunlight for the energy needed for photosynthesis process. From simple inorganic compound such as water, Carbon dioxide etc. organic compound like sugar, fat and protein are synthesized. [4] Therefore, the plant leaves can provide vital information about plants physiological condition. This can be done by analyzing spectral properties of plants. The analysis of reflectance, transmittance and absorption of radiation in leaves can give information about the plant's growth and adaption with surroundings. [5]

2.2 Relevance of Infrared Radiation to plants

Spectral radiation of different wavelength specifically in non-visible range like infrared region are constantly used in agricultural sector. It is possible to identify stressed and non-stressed plants by observing the changes in optical properties of individual leaves. The reflectivity of plant leaves that are diseased, senescenced, or stressed, under the presence of infrared radiation sometimes increase or sometimes decrease (advance stage of senescence). It is found that for white and dry leaves the reflectivity of infrared radiation is very high for which the reason lies in the internal cellular structure of leaves. But most of the reflection does not happen from the upper part of the leaves rather, the infrared radiation undergoes multiple reflection and refraction inside the leaf structure. However, the picture is quite different if water molecule and chlorophyll are present. In that case most of the radiation is absorbed inside the leaf. Therefore, we get low reflectance as well as low transmittance. [6]

2.3 Principal component analysis (PCA)

The PCA has been viewed to be one of the most useful and important results of applied linear algebra . Often, scientists are trying to figure out certain concepts or phenomena by measuring physical quantities in a system as in this study. However, the data set obtained can appear confusing, ambiguous or unwanted and so becomes difficult to analyze. [7]

The PCA is a technique that attempts to analyze these complex data sets in order to form a new set of orthogonal data called principal components by extracting the important information that reveals the underlying relationship connecting the data sets together. [7]

2.4 Average learning subspace method (ALSM)

The subspace method has been a vital tool in reduction of dimensions and concurrent classification of data sets. It works on the principle that patterns belonging to a class in a high dimensional vector forms a cluster which can be classified into a subspace class. It has gone through series of extensions due to its drawbacks. One previous drawback was that the subspace of one class is independent of other subspace classes (Class-featuring information compression) while another is that learned samples can be over-written by next learned samples (Learning Subspace Method) [8].

To overcome this problem, an iterative learning algorithm ALSM was introduced which employs the PCA technique with few modifications. In this practice, a learning vector is classified using ALSM. Whenever the vector is classified incorrectly to a wrong subspace, the correct subspace is rotated towards the misclassified vector and the incorrect subspace is rotated away from it [8]

$$C_k^{(i)} = C_{k-1}^{(i)} + \sum_{j \neq i} a^{(i,j)} C_k^{(i,j)} - \sum_{j \neq i} b^{(i,j)} C_k^{(j,i)}, \tag{2.1}$$

where C_k is a correlation matrix, k is the product of spectrum with its transpose, a and b are small constants, and i and j are the index notations of vectors. This is repeated for a set of learning vectors until the error is significantly minimized.

Chapter III

Measurements and calculations

In this chapter, the procedures and precautions taken to acquire the data from samples are discussed. Furthermore, a detailed explanation of how the code was developed in order to properly classify the samples will be discussed and finally the observations discovered in the process of classifications are highlighted.

3.1 Measurement procedure

The experiment was conducted using a Line Scanning Spectral Imaging System. It includes scanner 1x1.5 m and two cameras: visible V10 (400-1000 nm) and SWIR N25 (1000-2500 nm) which scan samples both in the visible and infrared range respectively. The light source used was stabilized for thirty minutes before measurements were taken. Samples used in this experiment were plants selected from different species of trees (Spruce, Pine, and Birch) from which spectral cubes containing reflectance information were acquired. The samples were carefully placed in a defined region equivalent to an A4 paper size and were made to pass under the SWIR N25 (1000-2500 nm). Focusing of the camera and software controlling the device had been properly set before measurements were taken. It was done such that each plant was scanned separately before they were mixed together and scanned again. Compensation for exposure times were calculated by the software and the experiment was carried out in a dark room to avoid samples being exposed to other light sources.

3.2 Developing the code

The development of the ALSM algorithm was done using Matlab and was divided into two stages: learning phase (Appendix A) and the testing phase (Appendix B). The learning phase was intended to train the program to correctly identify each sample to a satisfactory degree of accuracy while the testing phase aims to see how the program is able to convincingly distinguish each sample when mixed together. The codes in learning phase and testing phase use supporting codes to function, presented in Appendix C.

3.3 Learning phase

Reflectance value is located in a spectral cube containing three dimensional (3D) pixel information (x, y, λ) cube which was converted into two dimensional (2D) pixel information (x, λ) plane before calculations were done. The pixels of samples to be trained were normalized to their unit length vectors by dividing the pixels by their vector lengths to eliminate data overflow. Correlation matrices were calculated for the normalized data by multiplying with its transpose:

$$C_i = A_i \times A_i^T, \tag{3.1}$$

where A represents the normalized data, and $i = 1, 2, 3$ is a subscript that identified the sample used. In this study highest value of i was three since three plants were used. The result was a square matrix from which eigenvalues and its corresponding eigenvectors were calculated. The columns of the matrix are eigenvectors and the diagonal elements of the matrix are eigenvalues. Eigenvalues and eigenvectors were calculated in Matlab using the following command:

$$[V, D] = \mathrm{eig}(C), \tag{3.2}$$

where V and D are the eigenvectors and eigenvalues respectively and C is the correlation matrix of each sample calculated from Eq. 3.1. The largest eigenvalues with their corresponding eigenvectors were used to calculate the projection values of the used samples. It should be noted that the eigenvectors are orthogonal to one another and thus can be used to calculate its projection from its equivalent original data. The projection value P was calculated by multiplying the transpose of V by the spectrum of the sample:

$$P_i = \text{norm}(V_i^T \times S), \tag{3.3}$$

where P_i is the projection value of each sample and S is the spectrum of the sample. Norm in Eq. 3.3 refers to the normalization of data vector.

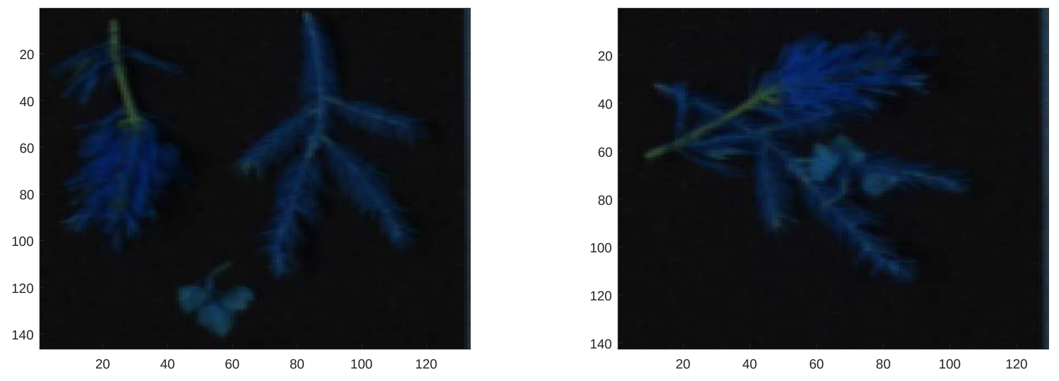
The code was developed such that when there is a correct classification of samples, it is assigned to its corresponding subspace class. However, when there is a misclassification, it should make modifications such a way the correlation matrix C is rotated towards the right subspace:

$$C_i = C_i + ak, \tag{3.4}$$

where a is a small coefficient and $k = S \times S^T$. This process was repeated for every pixel until a satisfactory or stable success rate was achieved.

3.4 Testing phase

When a high degree of success rate was achieved, the code was used to test if it could successfully distinguish each plant from two different samples: separated and mixed samples, presented in Figs. 3.1(a) and 3.1(b).



(a) Separated samples.

(b) Mixed samples.

Figure 3.1: RGB presentation of used test samples. Blue corresponds to 1040 nm, green to 1850 nm, and red to 2470 nm.

The code in the testing phase was developed such that the modified correlation matrices from the learning phase were used to get the projection values of the plants to be classified. Background pixels were set to zero to remove the effects of noise or other disturbances that could affect the classifications. A zero matrix equivalent to the size of the testing samples was created such that when each plant is successfully classified by its projection values, the zero matrix corresponding to the sample identified is set as 1. This is repeated until the pixels have all been classified. The resulting matrix forms a binary image of each classified plant separated from one another.

Chapter IV

Results and discussion

A significant portion of samples was carefully chosen from the original spectral cube of the plants. This contained enough information to be used in the learning phase. The data size used in the training was 19x8 pixels for spruce, 21x14 pixels for pine, and 20x8 pixels for birch. When there was a misclassification, rotation was done according to the Eq. (3.4).

Value of a was chosen to be 0.0026. A low value was chosen so that the rotation of the matrices could be coherent and gradually move towards their correct subspaces. High values will cause incoherence in the rotation of the matrices. During the iterative process, it was observed that at the fourth iteration all subspaces had been successfully classified and the success rate had reached 100 % as shown in Fig. 4.1.

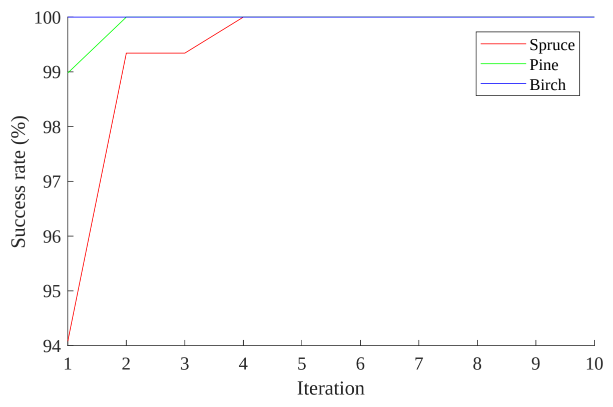
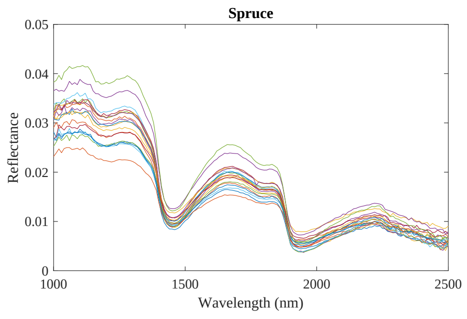
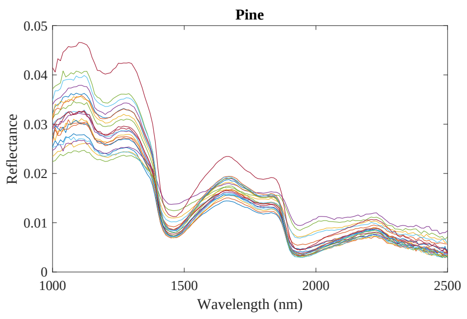


Figure 4.1: Success rate after 10 iterations.

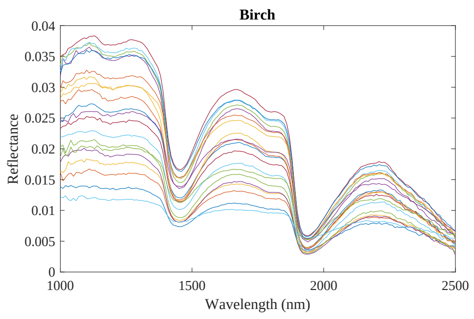
The success rate was very high and it took very low number of iterations to achieve this optimal classification. This could be because the experiment was conducted under infrared wavelengths. The reflectance spectra of samples are presented in Figs. 4.2(a)-(c). From these figures, it can be seen that spectra of Spruce and Pine are quite similar and propably accounted for some of the misclassifications in the learning phase. In the case of Birch, no misclassifications were observed, as can be predicted from Figs. 4.2(a)-(c).



(a) Spruce



(b) Pine

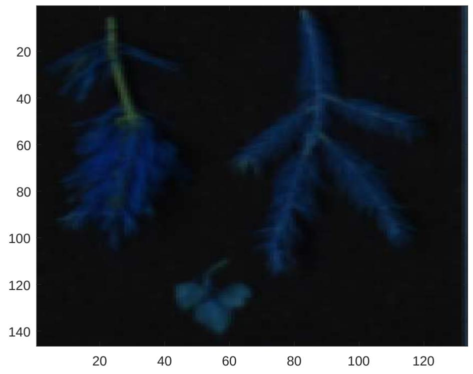


(c) Birch

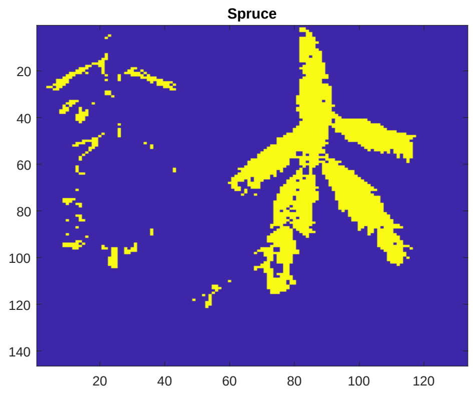
Figure 4.2: Normalized reflectance spectra of used samples.

In the testing phase, background pixels were set to zero in order to remove the effects of noise or other disturbances that could affect the classifications. The results of the testing phase are shown in Figs. 4.3(b)-(d) and 4.4(b)-(d). In these figures some misclassifications can be seen with Spruce and Pine. Intrestingly, Pine misclassified only a small amount of pixels that belong to Spruce. For Spruce the opposite is true: it misclassified a relatively large amount of needles of Pine to be Spruce. This is propably caused by similar spectral characteristics, presented in Figs. 4.2(a)-(c). In the case of Birch there seem to be only small random misclassifications of pixels but Birch is not classified to other plants.

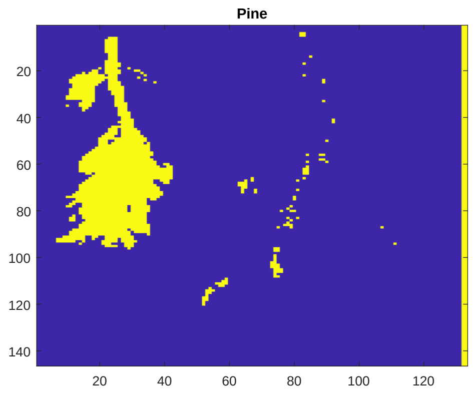
The differences between classifications in test samples were small. The classifications seem to be as accurate in mixed case (Figs. 4.4(a)-(d)) as well as in separated case (Figs. 4.3(a)-(d)). Some variation can be seen: Pine was classified more as Spruce in mixed case and the opposite in separated case but the variation was not significant.



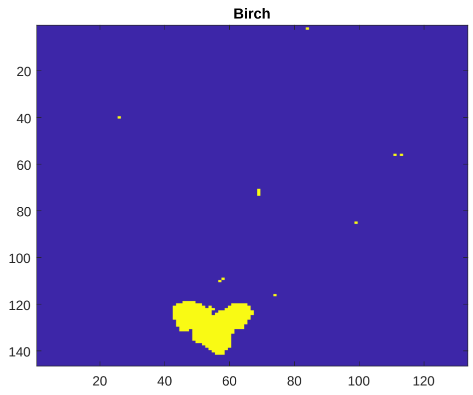
(a) Original image



(b) Classified as Spruce



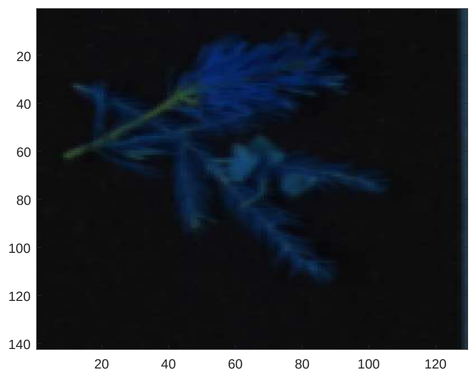
(c) Classified as Pine



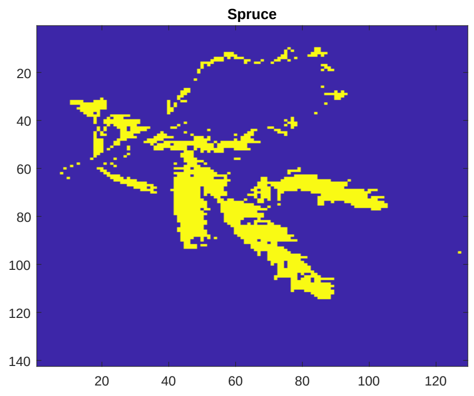
(d) Classified as Birch

Figure 4.3: Classification for Fig. 3.1(a)

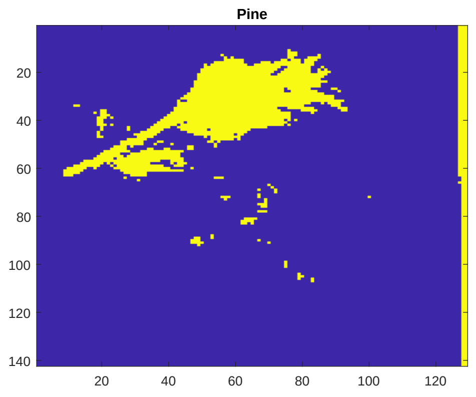
Some reasons for misclassification were cause by the spectral characteristics as discussed earlier but other reasons for misclassifications can be caused by small size of samples in the learning phase and low number of iterations.



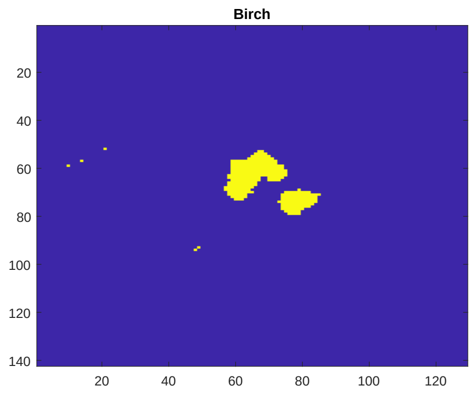
(a) Original image



(b) Classified as Spruce



(c) Classified as Pine



(d) Classified as Birch

Figure 4.4: Classification for Fig. 3.1(b)

Chapter V
Conclusions

Leaves, needles and twigs of Birch, Spruce and Pine were measured using a spectral line camera which works in infrared region of $1000 - 2500$ nm. Using spectral cubes from these measurements a code was taught to identify the used plants. The code used ALSM and PCA to teach itself how to differentiate different samples. The learning phase took only four iterations to achieve good results, propably due to differences in infrared spectrum of samples.

After the code was taught to identify plants it was tested using samples with all three plants, first separated from one another and later mixed with one another. The code succesfully identified the corresponding plants in both cases, but some misclassifications were seen in Spruce and Pine. This was propably due to their similar spectral characteristics. Birch was not misclassified at all. Some errors may be caused by low number of iterations and relatively small sample size used in learning phase. Despite the low amount of iterations in the learning phase the results are promising and this method can indeed be accurately used to identify plants in infrared region.

Bibliography

[1] R. Goodacre, “Explanatory analysis of spectroscopic data using machine learning of simple, interpretable rules,” *Vibrational Spectroscopy* **32**, 33–45 (2003).

[2] T. Howley, M. G. Madden, M.-L. OâĂŹConnell, and A. G. Ryder, “The effect of principal component analysis on machine learning accuracy with high-dimensional spectral data,” *Knowledge-Based Systems* **19**, 363–370 (2006).

[3] C. Cheng, J. Liu, C. Zhang, M. Cai, H. Wang, and W. Xiong, “An overview of infrared spectroscopy based on continuous wavelet transform combined with machine learning algorithms: application to chinese medicines, plant classification, and cancer diagnosis,” *Applied Spectroscopy Reviews* **45**, 148–164 (2010).

[4] D. M. Gates, H. J. Keegan, J. C. Schleter, and V. R. Weidner, “Spectral properties of plants,” *Applied optics* **4**, 11–20 (1965).

[5] G. A. Carter and A. K. Knapp, “Leaf optical properties in higher plants: linking spectral characteristics to stress and chlorophyll concentration,” *American journal of botany* **88**, 677–684 (2001).

[6] E. B. Knipling, “Physical and physiological basis for the reflectance of visible and near-infrared radiation from vegetation,” *Remote sensing of environment* **1**, 155–159 (1970).

[7] H. Abdi and L. J. Williams, “Principal component analysis,” *Wiley interdisciplinary reviews: computational statistics* **2**, 433–459 (2010).

[8] H. Bagan, Y. Yasuoka, T. Endo, X. Wang, and Z. Feng, "Classification of airborne hyperspectral data based on the average learning subspace method," *IEEE Geoscience and Remote Sensing Letters* **5**, 368–372 (2008).

Appendix A

Code for learning phase

```
% samples must be of the same size

%load sample data
load('sampleNew.mat');
% cell 1 = spruce
% cell 2 = pine
% cell 3 = birch

% get the number and size of samples
nSample = size(sampleNew); % get number of samples
sSample1 = size(sampleNew{1}); % get the size of sample data
sSample2 = size(sampleNew{2}); % get the size of sample data
sSample3 = size(sampleNew{3}); % get the size of sample data

% calculate the number of pixels on the sample
pixels1 = sSample1(1)*sSample1(2);
pixels2 = sSample2(1)*sSample2(2);
pixels3 = sSample3(1)*sSample3(2);

iClassificationSample = 1;  % Sample to be classified

%define new matrices to track classification

    class_matrix{1} = zeros(sSample1(1),sSample1(2));
    class_matrix{2} = zeros(sSample2(1),sSample2(2));
    class_matrix{3} = zeros(sSample3(1),sSample3(2));
    % cell 1 = spruce
    % cell 2 = pine
    % cell 3 = birch



iy =1; % define first y-coordinate
%load A, which is a x lambda plane of spectral cube
A1 = load_plane(iy,sampleNew{1}); %HAS TO BE NORMALIZED
A2 = load_plane(iy,sampleNew{2}); %HAS TO BE NORMALIZED
A3 = load_plane(iy,sampleNew{3}); %HAS TO BE NORMALIZED

% Normalize
A1 = A1./norm(A1);
A2 = A2./norm(A2);
A3 = A3./norm(A3);

% C vectors must be primed before entering the loop, otherwise we can not
% rotate them
%Calculate the correlation matrices C1, C2 and C3
C1 = A1*A1';
C2 = A2*A2';
C3 = A3*A3';

% set coefficients for k vectors
a = 0.0026;

ix = 1; % set first x coordinate
iteration = 1; % number of iterations done
stop = 10; % number of iterations you want
successRate = zeros(stop,nSample(2));
while iteration < stop+1
    sprintf('Starting iteration cycle %d/%d...',iteration,stop)
    misclassification(1) = 0; %reset misclassification variable
    misclassification(2) = 0; %reset misclassification variable
    misclassification(3) = 0; %reset misclassification variable
    iClassificationSample = 1;

    while iClassificationSample < 3+1
        % get the size of sample data
        sSample = size(sampleNew{iClassificationSample});
        iy = 1;
```

```
while iy < sSample(2)+1
       ix = 1;
    while ix < sSample(1)+1
        %while iClassificationSample < 3+1
        % create spectrum vector S
        S = load_px(ix,iy,sampleNew{iClassificationSample});
        %calculate eigenvalues for all C
            [V1 D1] = eig(C1);
            [V2 D2] = eig(C2);
            [V3 D3] = eig(C3);

        % Calculate P for sample to be classified
            P1 = norm(V1(:,end-4:end)'*S);
            P2 = norm(V2(:,end-4:end)'*S);
            P3 = norm(V3(:,end-4:end)'*S);
        % Choose highest P
            class = -1; %prime class variable, -1 = not classified
            if P1 > P2 % sample is not pine!
                if P1 > P3 % sample is spruce!
                    class = 1;
                else % sample is birch!
                    class = 3;
                end
            else % sample is not spruce!
                if P2 > P3 % sample is pine!
                    class = 2;
                else % sample is birch!
                    class = 3;
                end
            end
            % mark in which class the pixel was classified into
            class_matrix{iClassificationSample}(ix,iy) = class;

        % check classification and react to wrong classification
             if iClassificationSample ~= class
                %this loop triggers if classification is wrong
                 k = S*S'; % k is calculated from used data
                 if class == 1 % wrong class is 1
                     % this means we rotate the right class
                     % away from 1
                     if iClassificationSample == 2
                         % rotate C2, because right class was 2
                         C2 = C2 + a*k;
                     else
                         % rotate C3, because right class was 3
                         C3 = C3 + a*k;
                     end
                 elseif class == 2 % wrong class is 2
                     % this means we rotate the right class
                     % away from 2
                     if iClassificationSample == 1
                         % rotate C1, because right class was 1
                         C1 = C1 + a*k;
                     else
                         % rotate C3, because right class was 3
                         C3 = C3 + a*k;
                     end
                 else % wrong class is 3
                     % this means we rotate the right class
                     % away from 3
                    if iClassificationSample == 1
                        % rotate C1, because right class was 1
                         C1 = C1 + a*k;
                    else
                        % rotate C2, because right class was 2
                         C2 = C2 + a*k;
                     end
                 end
                 misclassification(iClassificationSample)
                 = misclassification(iClassificationSample)+1;
             end % end misclassification
        ix=ix+1; % change pixel
    end%ix
iy = iy+1;

end% iy
%change sample
iClassificationSample = iClassificationSample + 1;
```

```
    end%change class
    % calculate success rate for this iterations
    successRate(iteration,1) = iteration;   % iteration
    % spruce
    successRate(iteration,2) = (pixels1-misclassification(1))/pixels1;
    % pine
    successRate(iteration,3) = (pixels2-misclassification(2))/pixels2;
    % birch
    successRate(iteration,4) = (pixels3-misclassification(3))/pixels3;
    iteration = iteration+1;

end% change iteration
% plot success rate
figure; hold on
plot(successRate(:,1),successRate(:,2).*100,'r-') % spruce
plot(successRate(:,1),successRate(:,3).*100,'g-') % pine
plot(successRate(:,1),successRate(:,4).*100,'b-') % birch
legend('Spruce','Pine','Birch')
xlabel('Iteration')
ylabel('Success rate (%)')
hold off

% plot A vectors for spectral info
   figure;plot([1000:10:2500],A2);title('Pine');
   figure;plot([1000:10:2500],A1);title('Spruce');
   figure;plot([1000:10:2500],A3);title('Birch');
```

Appendix B

Code for testing phase

```
% load separate leaves
%load('127_reflectance_r.mat');
% load mixed leaves
load('128_reflectance_r.mat');
test = spectral_cube;
sSample = size(test); % get the size of sample data


%define new matrices to track classification
spruce = zeros(sSample(1),sSample(2));
pine = zeros(sSample(1),sSample(2));
birch = zeros(sSample(1),sSample(2));

%use C vectors from learning phase
[V1 D1] = eig(C1);
[V2 D2] = eig(C2);
[V3 D3] = eig(C3);
iy = 1; ix = 1;
%delete background data
   test = background(test);
while iy < sSample(2)+1
while ix < sSample(1)+1
    % create spectrum vectors
        S = load_px(ix,iy,test);
    % Check if pixel is background
    spectrumSum = 0;
    for n = 1:151
```

```
        spectrumSum = spectrumSum + S(n);
    end
    if spectrumSum ~= 0
        % Calculate P for sample to be classified
            P1 = norm(V1(:,end-4:end)'*S);
            P2 = norm(V2(:,end-4:end)'*S);
            P3 = norm(V3(:,end-4:end)'*S);
        % Choose highest P
            class = -1; %prime class variable, 0 = not classified
            if P1 > P2 % sample is not pine!
                %class = 1;
                if P1 > P3 % sample is spruce!
                    spruce(ix,iy) = 1;
                else % sample is birch!
                    birch(ix,iy) = 1;
                end
            else % sample is not spruce!
                %class = 2;
                if P2 > P3 % sample is pine!
                    pine(ix,iy) = 1;
                else % sample is birch!
                    birch(ix,iy) = 1;
            end
        end
    end
    ix=ix+1; % change pixel in x-direction
end
iy = iy+1; % change pixel in y-direction
ix = 1; % reset pixel in x-direction
end

%plot figures
figure;imagesc(spruce);title('Spruce');
figure;imagesc(pine);title('Pine');
figure;imagesc(birch);title('Birch');
```

Appendix C
Supporting codes

Remove background pixels

```
function [test] = background(test)

ix = 1;
iy= 1;
s = size(test);
%create new matrix for data
cubeNoBG = zeros(s(1),s(2),s(3));
while ix < s(1) + 1
    while iy < s(2) + 1
        pixel = load_px(ix,iy,test); % load pixel
        if sum(pixel)/151 > 0.08 % check average value of pixel
             % add data that is not background into the new matrix
            cubeNoBG(ix,iy,:) = test(ix,iy,:);
        end
        iy = iy + 1;
    end
    ix = ix + 1;
    iy = 1;
end

test = cubeNoBG;
```

Load $x - \lambda$ plane

```
function [plane] = load_plane(iy,spectral_cube)
%% Loads lambda - x plane from spectral cube
%define y-coordinate from spectral cube
 %iy = 1;

%load x-plane
ix = 1;

%read the size of cube
s = size(spectral_cube);

plane = zeros(s(1),151); %s(1) = size of x-axis
while ix < s(1)+1
    plane(ix,:) = load_px(ix,iy,spectral_cube);
    ix = ix+1;
end
plane = plane'; %plane was wrong way around so we flip it
```

Load pixel

```
%% This function takes the wavelength data for one pixel
function [spectra] = load_px(ix,iy,spectral_cube)
lambda = 1;
while lambda < 151+1
    spectra(lambda,1) = spectral_cube(ix,iy,lambda);
    lambda=lambda+1;
end
```