ENCM 509 - Laboratory #3

Kyle Derby MacInnis

October 7, 2015

Abstract

This laboratory focuses on clustering collected signature data into two classifications representing the null hypothesis and alternative hypothesis to determine whether given data has probable likelihood using the Expectation Maximization algorithm to perform the classification.

Introduction

This lab looks at statistical pattern recognition using both supervised and unsupervised training schemes. The lab uses signature data collected previously and focuses on two main parameters of interest: pressure and velocity.

The data is used to generate joint probability density functions (PDF) and then a log-likelihood calculation is performed to help cluster the data and determine the likelihood of being either part of the null hypothesis or the alternative. For the unsupervised learning, the expectation maximization (EM) algorithm is used without prior knowledge of the signature data being used.

Procedure

1. Load in Data from signatures into a structured array.
2. Calculated the Average pressure and velocity for each signature.
3. Calculate a Joint PDF of the signature data using the average velocity and pressure as the variables.
4. Using the PDF, calculate log-likelihood values for signatures in relation to either the Null(Authentic) or Alternative(Forged) hypothesis.
5. Use the EM algorithm to perform unsupervised classification into two clusters using the signature data.

Results

Supervised Learning

The following results are those obtained from the supervised learning stage and relate to steps 1-4:

Supervised Authentic Signature Distribution Calculation Results

Calculated Mean μ_a	176.3889	175.3852
Calculated Covariance	116.9406	114.8199
Σ_a	114.8199	115.5789

Supervised Forged Signature Distribution Calculation Results

Calculated Mean μ_f	186.7029	186.1780
Calculated Covariance	349.9792	331.1016
Σ_f	331.1016	316.8010

Calculated Distributions (Supervised)

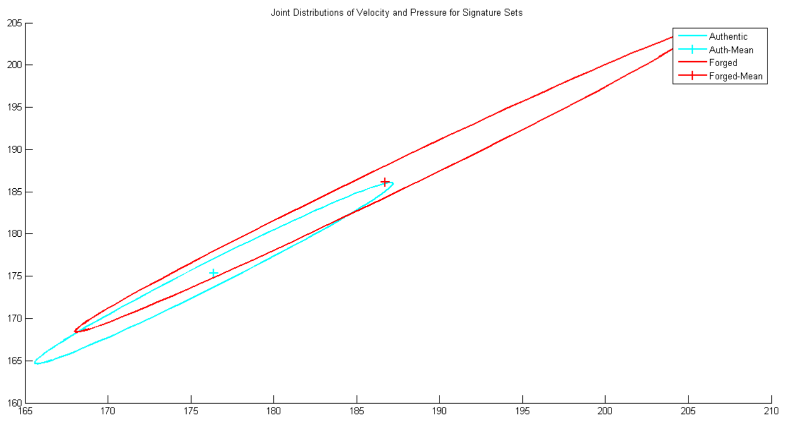

Figure 1: Calculated Distributions for both the Authentic and Forged Signatures

Log-like Values for Authentic Signature Distribution

Log-like Values (Auth)	Log-like value (Forg)
-6.9532	-14.8500
-5.9317	-5.9723
-5.6510	-7.5713
-6.1262	-6.1455
-5.6740	-10.5212
-6.0996	-5.6231
-6.0954	-5.8426
-6.5347	-6.6944
-5.7244	-5.9255
-5.8798	-5.8833

Authentic PDF Plots

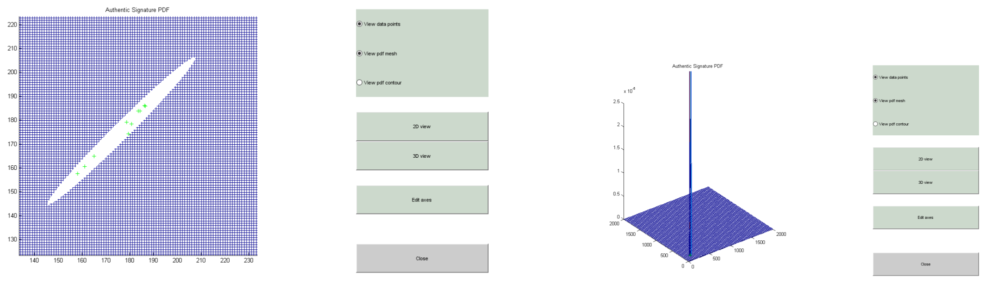

Figure 2: Authentic Signature PDF's in 2D(left) and 3D(right)

Log-like Values for Forged Signature Distribution

Log-like Values (Auth)	Log-like value (Forg)
-7.4153	-8.6800
-6.1249	-6.1249
-6.1857	-6.4070
-6.1324	-6.1340
-6.1707	-7.2532
-6.8258	-6.3358
-6.1301	-6.1291
-7.1401	-7.2473
-6.1501	-6.6836
-6.1265	-6.1263

Forged PDF Plots

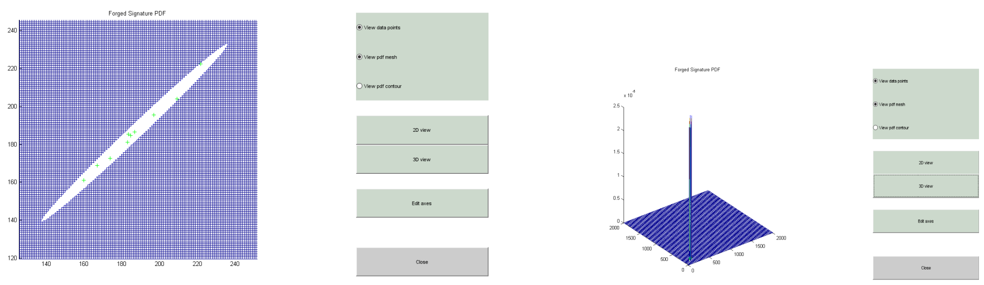

Figure 3: Forged Signature PDF's in 2D(left) and 3D(right)

Unsupervised Learning

The following uses the data collected in step 1, and performs the EM algorithm on the data to classify it into two clusters - representing the null(authentic) and alternative(forged) hypotheses:

Unsupervised Authentic Signature Distribution Calculation Results

Calculated Mean (EM) μ_a	177.4285	177.1506
Calculated Covariance (EM) Σ_a	120.7828	117.0408
	117.0408	114.5874

Unsupervised Forged Signature Distribution Calculation Results

Calculated Mean (EM) μ_f	202.2549	199.0475
Calculated Covariance (EM) Σ_f	305.6429	335.5859
	335.5859	372.4159

Expectation Maximization Run-Through

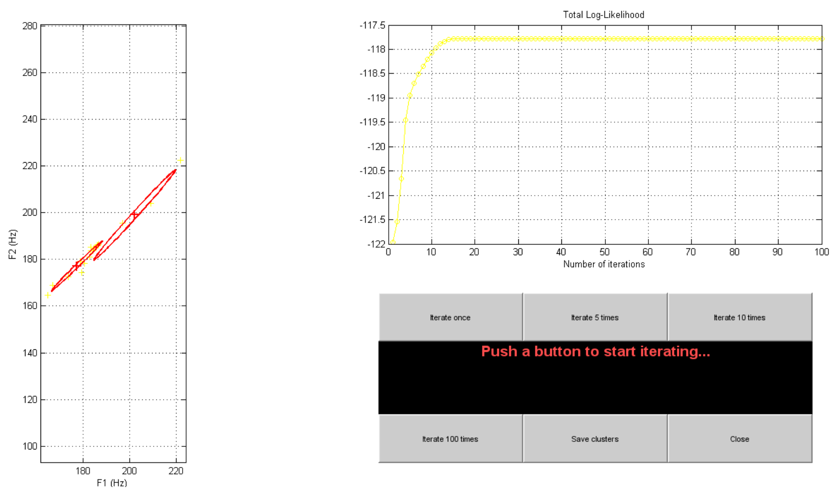

Figure 4: After 100 iterations the EM algorithm results in the following clusters.

Discussion

During the supervised learning phase, the calculated PDFs for the forged and signature data were distinct. The forged PDF overlapped a large portion of the Authentic PDF which is to be expected due the variance within the forged data, whereas the authentic signatures had a much more distinctive region and the variance was quite a bit lower. This look like it would result in a good FAR error response, whereas the forged data being so broad could result in a few FRR errors due to the larger variation and partial coverage of the authentic region, so it should be weighted less to compensate for its extra encapsulation.

Comparing the values found using the supervised learning, and those found using the unsupervised learning the following analysis of the data was found:

Authentic Signature Distribution Comparison

Mean (% Difference) μ_a	0.5%	1.0%
Covariance (% Difference)	3.3%	1.9%
Σ_a	1.9%	0.8%

Forged Signature Distribution Comparison

Mean (% Difference) μ_f	8.3%	6.9%
Covariance (% Difference)	12.7%	1.3%
Σ_f	1.3%	17.6%

As can be seen, the results are pretty similar between both the supervised and unsupervised learning routines. However, it should noted that the percent error between the authentic signature data is much lower, than the percent error found for the forged. This is not surprising however, since it would be expected that the forged data would have less similarities between them than that of the authentic data and thus could pose a problem for correct assessment. This suggests that the EM algorithm may be better suited for reducing FAR errors rather than for reducing FRR errors as it could better discern authentic signature data than that of the forged signatures. The initial *a-priori* values were 0.5 per cluster and after 100 iterations the final *a-priori* values found from the EM were 0.8342 for authentic and 0.1658 for the forged signatures. This goes to show that the EM was pretty good at getting a similar distributions to the supervised learning method, but with this small amount of data was not enough for it to properly discern the correct number of authentic signatures which was in fact 50% of the total sample set. This weighting however does help reduce the FRR errors in making a full decision.

Remarks on the Lab

This lab was quite good because of the visual nature, but I feel that more focus on building on the errors and relationship between the second lab and this lab as I feel it could help to make the foundations more concrete.

Appendix A: MATLAB Code

ENCM509_Lab3.m

```
% ENCM 509 Laboratory #3
%
%   Author: Kyle Derby MacInnis
%
%   Date: October 1, 2015

% Close all Figures
close all;

% Load Signature Arrays
sigAuth = load('sigAUTH.mat');
sigForg = load('sigFORG.mat');
% Convert to Arrays
auth = struct2array(sigAuth);
forg = struct2array(sigForg);

% Sample size
N_a = size(auth,2);
N_f = size(forg,2);

%% PART I - Calculate Average Pressure and Velocity

% Calculate Velocity and pressure Data
a = avgSigVel(auth, N_a);
f = avgSigVel(forg, N_f);

% Plot the data as gaussian clusters (Auth)
figure('name','Gaussian Cluster Plots - Part 1');
hold on;
mu_a = mean(a);
sigma_a = cov(a);
plotgaus(mu_a, sigma_a, [0 1 1]);
% (Forged)
mu_f = mean(f);
sigma_f = cov(f);
plotgaus(mu_f, sigma_f, [1 0 0]);
% Put Information
title('Joint Distributions of Velocity and Pressure for Signature Sets');
legend('Authentic','Auth-Mean','Forged','Forged-Mean');
hold off;

% Display Stats
disp('Authentic Data')
fprintf('## Authentic Mean:\n');
disp(mu_a);
fprintf('## Authentic Covariance:\n');
disp(sigma_a);
disp('Forged Data')
fprintf('## Forged Mean:\n');
disp(mu_f)
fprintf('## Forged Covariance:\n');
disp(sigma_f)

%% PART II - Gaussian Log Like Calculations
% Calculate Log Plots
N = N_a + N_f;
% Calculate Initial Priori Probabilities
P_a = N_a/N;
P_f = N_f/N;

% Calculate Loglike Values
loglikeAuth_a = calcLogLike(a,N_a,P_a,mu_a,sigma_a);    % Auth to Auth
loglikeAuth_f = calcLogLike(f,N_f,P_a,mu_a,sigma_a);    % Forg to Auth
loglikeForg_a = calcLogLike(a,N_a,P_f,mu_f,sigma_f);    % Auth to Forg
loglikeForg_f = calcLogLike(f,N_f,P_f,mu_f,sigma_f);    % Forg to Forg

% Plot Gaussian Plots
gausplot(a,mu_a,sigma_a,'Authentic');%,.5*min(a(:,1)):1.5*max(a(:,1)),.5*min(a(:,2))
    :1.5*max(a(:,2)))
```

```
gausplot(f,mu_f,sigma_f,'Forged');%,.5*min(f(:,1)):1.5*max(f(:,1)),.5*min(f(:,2)):1.5*
    max(f(:,2)))

% display stats
fprintf('Authentic Log-Like Values for Auth:\n');
disp(loglikeAuth_a);
fprintf('Forged Log-Like Values for Auth:\n');
disp(loglikeAuth_f);
fprintf('Authentic Log-Like Values for Forged:\n');
disp(loglikeForg_a);
fprintf('Forged Log-Like Values for Forged:\n');
disp(loglikeForg_f);

%% PART III - Unsupervised Training
% Create Data Set from pressure and velocity
dataSet = [a; f];
dataMeans = {mu_a, mu_f};
dataVars = {sigma_a, sigma_f};
% Use EM algorithm to define clusters (2 Clusters)
emalgo(dataSet, dataMeans, dataVars);
```

combine.m

```
function combine(filename, num, auth)
%function combine(filename, num, auth)
%
%   Combines Selected .Mat Files of the form filenameX.mat
%   where X is the num variable passed from 1-num.
%
%   auth is set to 0 for true, or any other value for false
%

    sigAll = cell(1,num);
    sAuth = cell(1,num);
    for i=1:num
        sigAll{i} = strcat(filename,num2str(i),'.mat');
        s(i) = load(sigAll{i});
        sAuth{i} = double(struct2array(s(i))');
    end

    if auth == 0
        save sigAUTH sAuth;
    else
        sForg = sAuth;
        save sigFORG sForg;
    end
```

calcLogLike.m

```
function loglike = calcLogLike(arr,N,p,mu,sigma)
% function loglike = calcLogLike(arr,N,p,mu,sigma)
%
%   Returns a vector of loglike values based on
%   the passed data and the given distribution
%   information. The distribution used is gaussian
%

% Initialize Empty Loglike vector
loglike = zeros(N,1);

% Loop through Array
for i = 1:N
    loglike(i) = gloglike(arr(i), mu, sigma)+log(p);
end
```

avgSigVel.m

```
function arr = avgSigVel(data_arr, N)
% function arr = avgSigVel(data_arr, N)
%
%   Performs Calculations on passed array of N structs to
%   calculate the average velocity for each.
%   Return an array of [prs, vel]

% Generate return array
arr = zeros(N, 2);

% Loop through authentic
for i = 1:N
    % Select Signature Sample (and transpose)
    arr_i = data_arr{i};

    % Extract Variables of interest
    x_i = arr_i(1,:);
    y_i = arr_i(2,:);
    prs_i = arr_i(3,:);
    time_i = arr_i(4,:);

    % Initialize Calculated variables
    xVel_i = zeros(1,max(length(x_i)));
    yVel_i = zeros(1,max(length(y_i)));

    % Calculate velocity
    for j=2:max(length(xVel_i))
        xVel_i(j) = (x_i(j)-x_i(j-1))/(time_i(j) - (time_i(j-1)));
        yVel_i(j) = (y_i(j)-y_i(j-1))/(time_i(j) - (time_i(j-1)));
    end

    % replace time with newly calculated velocity
    arr_i(4,:) = sqrt(xVel_i.^2 + yVel_i.^2);

    % Calculate average pressure and velocity
    arr(i,1) = mean(arr_i(:,3)); % Pressure
    arr(i,2) = mean(arr_i(:,4)); % Velocity
end
```