Pattern Classification

Using Bayesian Classifier; Least Mean Square; and Weiner Filter Algorithm

Nishant Bansal
EEE-511; Project #2
09/19/2014

***Abstract*— In this study, pattern classification results of different classifiers are analyzed. The strategy is to train the classifiers with two Gaussian 2-D clusters of 1000 data points each and classify a Gaussian test dataset of 200 data points to compute the confusion matrix and accuracy for each classifier. The clusters center at (0, 0), and (2.5, 0) and have unit variance. Classification results are plotted and discussed.**

***Keywords*— Machine Learning; Artificial Neural Netwroks; Bayesian classifier; Least Mean Square; Weiner Filter.**

I. Introduction

The objective of this study is to build different classifiers from two Gaussian 2-D clusters of 1000 points each. The variance of both clusters is one and the center of first cluster (mentioned as Class A) is (0, 0) and of second cluster (mentioned as Class B) is (2.5, 0). Using these classifiers then a test dataset consisting of 100 data points from each cluster (Class) is classified. The training dataset is shown in figure 1.

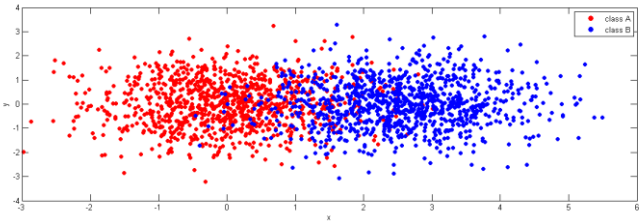

Fig. 1: Plot of the 2 Gaussian 2-D clusters (training dataset)

The confusion matrix and accuracy for each classifier is computed to analyze the classifier. The confusion matrix is a tabular layout which enables easy visualization of the performance of a classifier (algorithm, in general). Figure 2 shows the layout of the confusion matrix for 2 classes.

		Predicted Class	
		Class A	Class B
Actual Class	Class A	c_{11}	c_{12}
	Class B	c_{21}	c_{22}

Fig. 2: Tabular representation of a 2x2 confusion matrix

The 1st cell (c_{11}, also called true positives) is the count of data points which belong to Class A and are classified as Class A whereas c_{12} (also referred as false negatives) is the number of data points classified as Class B but actually are of Class A. The confusion matrix for an ideal classifier will be an identity matrix as it will not have any misclassification. The accuracy is defined as ratio of total number of data points correctly classified to the total number of data points.

$$Accuracy\ (a) = (c_{11} + c_{22}) / (c_{11} + c_{12} + c_{21} + c_{22})$$

II. Classifiers

A. Bayesian Classifier

Bayesian Classifier, also known as Bayes Classifier, is a probabilistic classifier based on Bayes' Theorem. It defines a average risk function (R) which depends on the prior probability (p_i) of X in H_i; cost (C_{ij}) of classification of X to H_i with true class as H_j; and conditional probability distribution function ($P_x(X/C_j)$) of X drawn from H_i.

$$R = c_{11}.p_1 + c_{22}.p_2 +$$
$$\int_{H1} [p_2(c_{12}-c_{22}).P_x(X/c2) - p_1(c_{21}-c_{11}).P_x(X/c_1)]dx$$

A likelihood ratio ($\wedge(x)$) and threshold (ξ) are also defined. If the likelihood ration is more than threshold than the data points belongs to Class A or else it belongs to Class B.

$$\wedge(x) = P_x(X/c_1)/\ P_x(X/c2)$$
$$\xi = p_2(c_{12}-c_{22})/\ p_1(c_{21}-c_{11})$$

For Gaussian clusters like our case the likelihood ratio is simplified in terms of means (μ_1 and μ_2) and covariance matrix (C). Considering $p_1 = p_2 = ½$; $c_{11} = c_{22} = 0$; $c_{12} = c_{21}$, the equation is simplified as follows:

$$log(\wedge(x)) = (\mu_1 - \mu_2)^T.C^{-1}.X + ½(\mu_2^T.C^{-1}.\mu_2 - \mu_1^T.C^{-1}.\mu_1)$$
$$and,\ log(\xi) = 0$$
$$where,\ C = E[(x - \mu_1)(x - \mu_1)^T]$$

This equation is equivalent to $y = w^T.x + b$.

$$y = log(\wedge(x));$$
$$w^T = (\mu_1 - \mu_2)^T.C^{-1};\ and$$
$$b = ½(\mu_2^T.C^{-1}.\mu_2 - \mu_1^T.C^{-1}.\mu_1)$$

The confusion matrix representing the classification results from Bayesian classifier for the dataset used in this studied is shown in table 1. For better visualization, the matrix is row normalized.

Table 1: Resultant confusion matrix for Bayesian Classifier

		Predicted Class	
		Class A	Class B
Actual Class	Class A	0.91	0.09
	Class B	0.11	0.89

The accuracy thus computed is 90%. The figure 3 shows the classification of the data points. The red and blue dots represent correct classification (c_{11} and c_{22}) of data points whereas the '+' represent the misclassification (c_{12} and c_{21}) of data points.

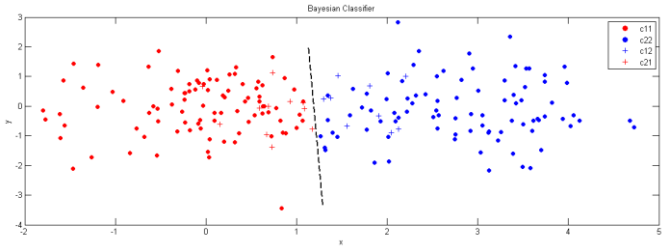

Fig. 3: Classification done by Bayesian Classifier. The line shows the decision rule learnt by the classifier for prediction. The '.' symbolizes correct classification whereas '+' symbolizes misclassification.

To get a better idea about the accuracy of this machine, it was run for 1000 iterations and the average accuracy was computed.

Avergae Accuracy = 89.54%

B. *Least Mean Square*

Least Mean Square (LMS) filter is an adaptive filter which works on the stochastic gradient descent method such that the filter adapts according to the error at the current iteration or time. It was invented by Bernard Widrow, Stanford University professor, in 1960. The structure of a LMS system is shown in Figure 4. As shown in the figure the error computed in each iteration is used as a feedback for the next iteration.

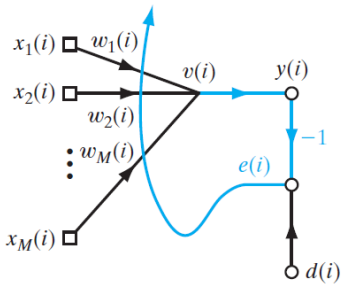

Fig. 4: An Adaptive system with error as a feedback

It introduces an instantaneous cost function $\xi(\hat{w})$ which depends on the error e(n). Let the desired label/classification be d(n), the data points be x(n), the instantaneous Weiner synaptic weights be $\hat{w}(n)$, and η be the learning rate, then for a linear neuron,

$e(n) = d(n) - x^T(n).\hat{w}(n)$

$\xi(\hat{w}) = \frac{1}{2}.e^2(n)$

then, $\partial\xi(\hat{w})/\partial\hat{w}(n) = -x(n).e(n) = \hat{g}(n)$

where, $\hat{g}(n)$ is the instantaneous gradient

So, $\hat{w}(n+1) = \hat{w}(n) + \eta.x(n).e(n)$

The algorithm adjusts the synaptic weights based on the instantaneous error to minimize the total error. The learning terminates when error change for the change in the weights is nearly constant, that is the change in weights doesn't really effect the change in error. So, the classification of the test dataset t(n) will be given by:

$y(n) = signum(t(n).w)$

where, signum function is

$signum(x) = 1$ if $x > 0$, and -1 otherwise.

The LMS when used for classification of our two Gaussian 2-D cluster dataset gave an accuracy of 88.5%. The average accuracy for 1000 iterations is 87.10%. The corresponding confusion matrix is shown in table 2.

Table 2: Resultant confusion matrix for LMS

		Predicted Class	
		Class A	Class B
Actual Class	Class A	0.91	0.09
	Class B	0.14	0.86

The classification thus obtained by LMS is visually shown in figure 5. The red and blue dots represent correct classification (c_{11} and c_{22}) of data points whereas the '+' represent the misclassification (c_{12} and c_{21}) of data points.

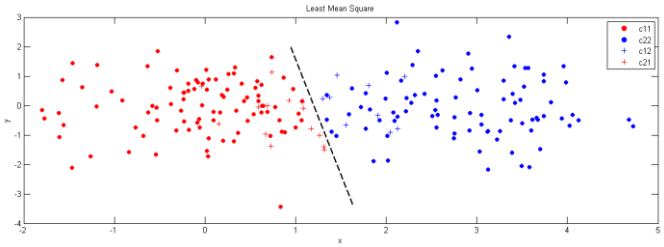

Fig. 5: Classification done by LMS. The line shows the decision rule learnt by the classifier for prediction. The '.' symbolizes correct classification whereas '+' symbolizes misclassification.

A part of the learning curve for LMS is shown in figure 6. It shows that the mean square error is decreasing. For this study the algorithm ran for all the data points.

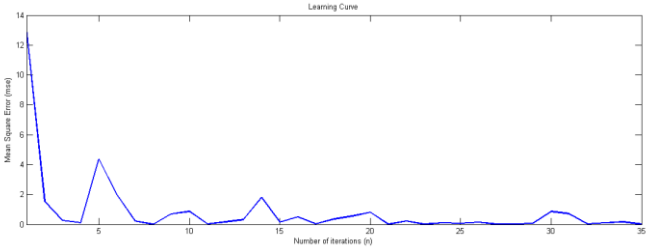

Fig 6. Learning Curve for LMS

C. *Weiner Filter*

The idea behind Weiner Filter is also based on minimizing the least mean square error. It is a batch learning algorithm which uses Gauss-Newton method to converge in a single iteration to find the weights for classification. Given x(n) be the data points; d(n) be the desired label/classification; w(n) be the weights; then the error e(n) will be

$$e(n) = d(n) - X(n).w(n)$$
$$where;\ X(n) = [x(1), x(2)\ldots x(n)]^T$$
$$then;\ \Delta e(n) = -X(n)^T$$
$$we\ get;\ w(n+1) = (X^T(n).X(n))^{-1}.X^T(n).d(n)$$

The term $(X^T(n).X(n))^{-1}.X^T(n)$ is called pseudoinverse of the data matrix X(n) and is represented by $X^+(n)$.

$$So,\ w(n+1) = X^+(n).d(n)$$

Similar to LMS these weights can be used to compute the predicted classification of the test dataset. The classification results for the dataset used in this study are shown below in the Figure 3 in the form of a row normalized confusion matrix. The accuracy obtained by Weiner Filter classification is 90.0%. The average accuracy for 1000 iterations is computed to be 89.51%.

Table 3: Resultant confusion matrix for Weiner Filter

		Predicted Class	
		Class A	Class B
Actual Class	Class A	0.93	0.07
	Class B	0.13	0.87

And figure 7 visualizes the classification in the form of a scatter plot. The line shows the decision rule thus obtained by the filter. The red and blue dots represent correct classification (c_{11} and c_{22}) of data points whereas the '+' represent the misclassification (c_{12} and c_{21}) of data points.

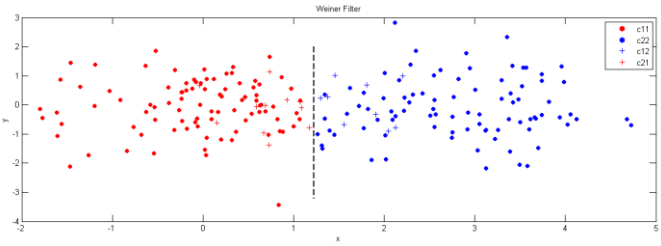

Fig. 7: Classification done by Weiner Filter. The line shows the decision rule learnt by the classifier for prediction. The '.' symbolizes correct classification whereas '+' symbolizes misclassification.

III. Conclusion

Each classifier technique used in this study converged and successfully classified the test data set. The data used is not linearly separable but still all the classifiers converged as compared to the Perceptron. The average accuracy for each classifier gives a good idea about the performance of each classifier. The Bayesian classifier and Weiner filter have approx accuracy of 89.5% whereas the LMS algorithm has a lower average accuracy of approximately 87%.

ACKNOWLEDGMENT

I would like to thank Professor Dr. Jennie Si to impart in depth knowledge in these topics and for the opportunity to work on it. I would also like to thank my team mate Rachit Agarwal for open discussions which helped me understand the concepts in more details.

REFERENCES

[1] S. Haykin, "Neural Networks and Learning Machines (3rd edition)," Upper Saddle River, NJ: Pearson Education, 2009.

[2] www.wikipedia.org

MATLAB CODE

```
% the main function for this study.
function [ cm_bayes, cm_lms, cm_weiner, bayes_acc, lms_acc, weiner_acc, mse ] = hw2()

    plot_flag = 0;
    meu = [ 0 2.5; 0 0 ];
    sigma = [ 1 0; 0 1 ];
    dataset = dataset_2( meu, sigma, 1000, -plot_flag );
    testset = dataset_2( meu, sigma, 100, plot_flag );

    % classification using Bayes
    y_bayes = Bayes_classifier(  dataset, testset(:, 1:2), meu);
    cm_bayes = confusion_matrix( testset(:, 3), y_bayes);
    bayes_acc = accuracy(cm_bayes);

    %plot
    if plot_flag == 1
        plot_classification(testset, y_bayes, 2);
    end

    % classification using LMS
    [ y_lms, mse ] = LMS_classifier( dataset, testset(:, 1:2), 0.05 );
    cm_lms = confusion_matrix( testset(:, 3), y_lms);
    lms_acc = accuracy(cm_lms);

    if plot_flag == 1
        plot_classification(testset, y_lms, 3);
    end

    % classification using Weiner
    [ y_weiner ] = Weiner_classifier( dataset, testset(:, 1:2) );
    cm_weiner = confusion_matrix( testset(:, 3), y_weiner);
    weiner_acc = accuracy(cm_weiner);

    if plot_flag == 1
        plot_classification(testset, y_weiner, 4);
    end
end

% it generates random 2D Gaussian data set
function [ dataset ] = dataset_2( meu, sigma, count, plot)
    % getting Gaussian Clusters
    cluster1 = mvnrnd(meu(:, 1), sigma, count);
    cluster2 = mvnrnd(meu(:, 2), sigma, count);
    tempset = [ cluster1 ones(count, 1); cluster2 -ones(count, 1) ];
    % randomly shuffling done
    dataset = tempset(randperm(2*count), :);
    if plot == -1
        figure(1);
        scatter(cluster1(:,1), cluster1(:,2), 25, 'r', 'fill');
        hold on;
        scatter(cluster2(:,1), cluster2(:,2), 25, 'b', 'fill');
        legend('class A', 'class B');
    end
end
```

```
% Bayesian Classifier
function [ y ] = Bayes_classifier(  dataset, testset, meu )

    x = dataset(:, 1:2);
    d = dataset(:, 3);
    meu1 = meu(:,1);
    meu2 = meu(:,2);
    ind = find(d == 1);
    C = cov(x(ind, :));
    w = ((meu1 - meu2)'/C)';
    b = ((meu2'/C)*meu2 - (meu1'/C)*meu1)/2;

    y = sign(testset*w + b);
end

% LMS Classifier
function [ y, mse ] = LMS_classifier( dataset, testset, rate )

    n = size(dataset, 1);
    x = dataset(:, 1:2);
    x = [ones(n, 1) x];
    d = dataset(:, 3);
    w = zeros(n, 3);
    mse = zeros(n, 1);
    w(1, :) = 1;
    for i = 1:n-1
        e = ( d(i,1) - x(i, :)*w(i, :)' );
        mse(i,1) = (1/2)*(e^2);
        w(i+1, :) = w(i, :) + rate*x(i, :)*e;
    end

    testset = [ ones(size(testset, 1), 1) testset ];
    y = sign(testset*w(n,:)');
end

% Weiner Filter
function [ y ] = Weiner_classifier( dataset, testset )

    x = dataset(:, 1:2);
    x = [ones(size(dataset, 1), 1) x];
    d = dataset(:, 3);
    rx = x*x';
    rdx = x'*d;
    w = (inv(x'*x)*x')*d;

    testset = [ ones(size(testset, 1), 1) testset ];
    y = sign(testset*w);
end

% It computes the accuracy for given confusion matrix
function [ accuracy ] = accuracy( confusionMatix )

    c = confusionMatix;
    accuracy = (c(1,1) + c(2,2))/ (c(1,1) + c(1,2) + c(2,1) + c(2,2));
end
```

```
% determines the confusion matrix for classified data against the ground truth
function [ c ] = confusion_matrix( desired, classified )

    c = zeros(2);
    for i = 1:size(desired,1)
        if desired(i) == classified(i)
            if desired(i) == 1
                c(1, 1) = c(1, 1) + 1;
            else
                c(2, 2) = c(2, 2) + 1;
            end
        else
            if desired(i) == 1
                c(1, 2) = c(1, 2) + 1;
            else
                c(2, 1) = c(2, 1) + 1;
            end
        end
    end
    c(1, :) = c(1, :)./(c(1,1) + c(1,2));
    c(2, :) = c(2, :)./(c(2,1) + c(2,2));
end

% plots the data points according to classification and actual labels.
function [  ] = plot_classification( testset, y, fig )

    d = testset(:, 3);
    figure(fig);
    hold on;
    test = y + d;
    class11 = find(test == 2);
    scatter(testset(class11,1), testset(class11,2), 25, 'r', 'fill');
    hold on;
    class22 = find(test == -2);
    scatter(testset(class22,1), testset(class22,2), 25, 'b', 'fill');

    test = d - y;
    class12 = find(test == 2);
    scatter(testset(class12,1), testset(class12,2), 50, 'b', '+');
    class21 = find(test == -2);
    scatter(testset(class21,1), testset(class21,2), 50, 'r', '+');

    legend('c11', 'c22', 'c12', 'c21');
end

% gets the average accuracy over a certain iterations
function [ avg_bayes, avg_lms, avg_weiner ] = average_accuracy( iterations )
    a_bayes = zeros(iterations, 1);
    a_lms = zeros(iterations, 1);
    a_weiner = zeros(iterations, 1);
    for i = 1:iterations
        [ ~, ~, ~, a_bayes(i), a_lms(i), a_weiner(i), ~] = hw2();
    end
    avg_bayes = mean(a_bayes);
    avg_lms = mean(a_lms);
    avg_weiner = mean(a_weiner);
end
```