Tejas G. Mahale

Linear Classification

CSE 583/EE552 Pattern Recognition

Abstract :

In supervised learning methods, a classifier is a systematic approach for building classification models from input dataset[1]. If classification is based on linear combination characteristics or parameters, it is linear classification. The goal of linear classifier is use an object's characteristics to identify which class or group it belongs to[2]. In this project, we are comparing different classification techniques like least square classification and fisher's discriminant with KNN classifier. The least-squares approach gives an exact closed-form solution for the discriminant function parameters[3]. On other hand we used fisher's discriminator to reduce dimensionality of given dataset. With help of KNN classifier we predicted the class of reduced dimension dataset. With problem of 'Curse of dimensionality' for datasets having not enough examples, we expect fisher's linear discriminant analysis could help us to improve class-separability in lower dimensions. For this project, we are using wine, wallpaper, taiji datasets which has good combinations of features and classes.

Table of content:

Topic	page Number
1 Introduction	2
1.1 Discriminant Functions	3
1.2 Least Square Classifier	6
1.3 Fisher's Linear Discriminant	8
1.4 K nearest Neighbor	10
2 Approach	12
2.1 Data Used	12
2.2.1 Mathematics for Least square classifier	13
2.2.2 Mathematics for Fisher's Linear Discriminant	16
2.2.3 Mathematics for KNN	19
2.3 Confusion and Classification Matrix	20
2.4 Matlab Functions	21
3 Results	22
3.1 Least square Classifier	22
3.2 Fisher discriminator & KNN classifier	28
4. Conclusion	33
5. References	34

1.Introduction

The goal of classification is to take input vector x and assign it to k different classes C_k such that k = 1,..,k. In case of linear classification, if you consider visualization, the input space is divided into decision boundaries which are linear functions of input vector x and hence defined by $(D-1)$ - hyperplanes within the D-dimensional input space[3].

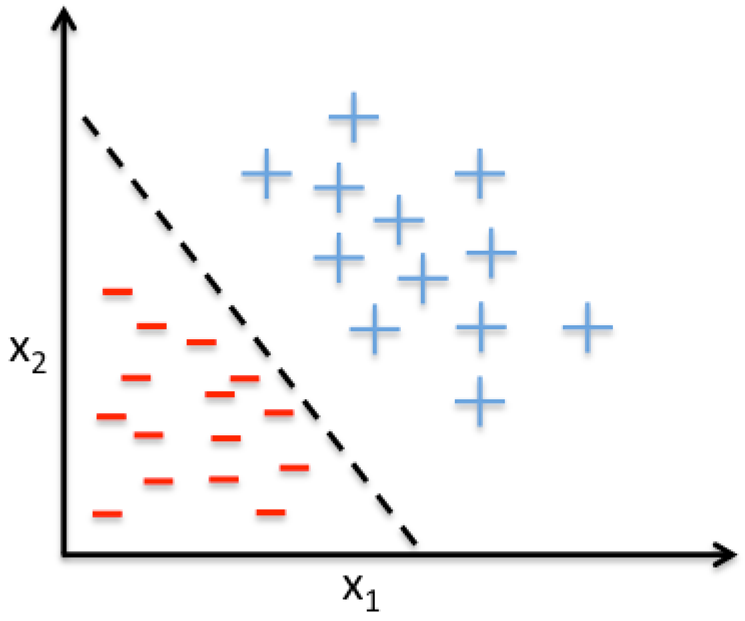


Figure 1.1 Linear classification

Figure 1 show linear classification for two classes $+\ and\ -$. We have to note that this is also example of two dimensional classification as we used two features only(X1 and X2). In reality data set may consist of multiple feature and multiple classes.

As output labels in classification are in discrete form, for dataset with more than two classes, we are going to use 1 to K encoded target vector where k is number of classes. Suppose class label for particular example m is 5 then it 1:k encoding is given by [0; 0; 0; 0; 1] ie. 5th row of mth column is 1 other elements in that column are 0.

$$\begin{array}{ccc} Y_k & & Y_K \\ \begin{bmatrix} 2 \\ 5 \\ 1 \\ 3 \end{bmatrix} & = & \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

1.1 Discriminant Functions

Discriminant function is function with several variates used to assign items into two or more classes. In our case discriminant function takes input x vector and labels each example to available classes(C_k). We will analyse discriminant function for two classes and more than two classes separately.

A) Two classes:

The simplest linear discriminator function is given by $Y(x) = W^T * X + W_0$.

Here X is input vector consist of features of number of examples x, W^T is transpose of weight vector, and W is a bias vector. In this case The decision boundary, Y(x)= 0, is a (D-1)-dimensional hyperplane within the D-dimensional input space.

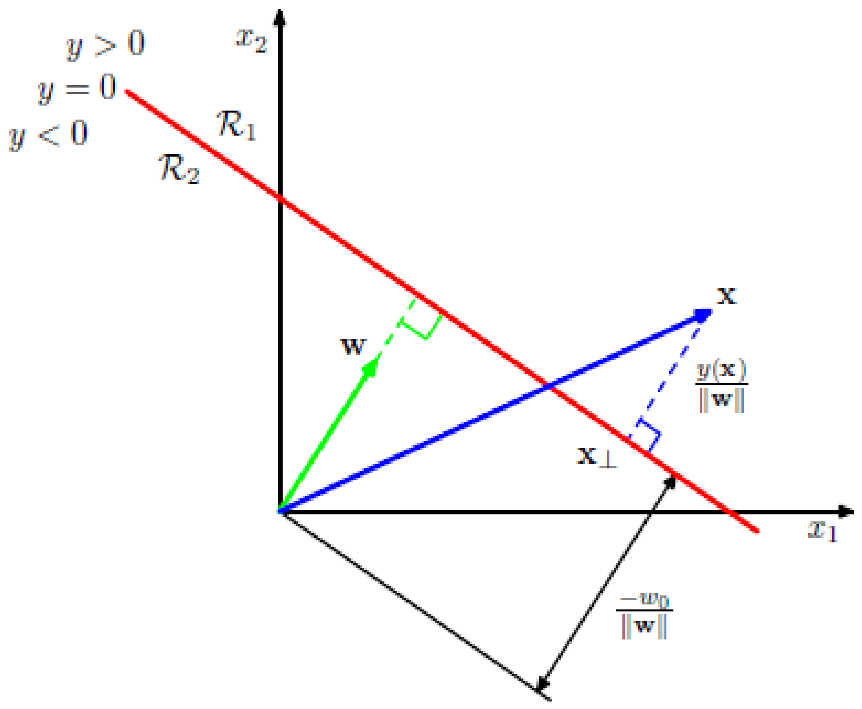
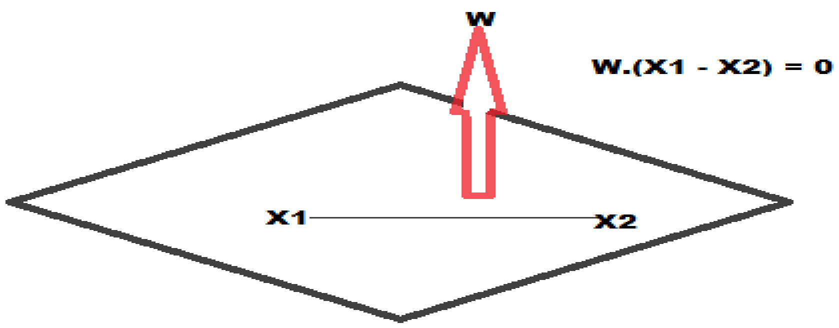


Fig. 1.1.1 Linear classifier for 2 classes from Bishop

In fig.1.1.1, red line separates two classes with regions R1 and R2. If point is in region R2, it will have class 2 label and if point satisfies y<0 condition, it will have class label 1. More intuition from this figure can be drawn. First of all, weight vector W is orthogonal to vector lying to decision surface. Second point is that, W_0 is shortest distance from origin to surface which determines location of decision surface.



$$\frac{W^T * X}{||W||} = \frac{-W_0}{||W||}$$ -------------- Eq 1.1.1

Consider arbitrary point x and its orthogonal projection on decision surface is x_t and Y(x) gives a signed measure of the perpendicular distance r of the point **x** from the decision surface. Then,

$$\mathrm{x} = x_t + r\frac{W}{||W||}$$

Multiplying both sides of this result by W^T and adding W_0,

$$W^T\mathrm{x} + W_0 = W^T x_t + W_0 + W^T r\frac{W}{||W||}$$

making use of $Y(\mathbf{x}) = W^T\mathbf{x} + W_0$, and $Y(x_t) = W^T x_t + W_0 = 0$, we have,

$$\mathrm{r} = \frac{\mathrm{Y(x)}}{||\mathrm{W}||}$$ -----------------Eqn 1.1.2

This result is illustrated in Fig. 1.1.1 Linear classifier for 2 classes.

B) More than two classes:

In this case, we can't simply follow K- class discriminant which uses K-1 classifiers each of which solves a two-class problem of separating points in a particular class C_k from points not in that class[3].

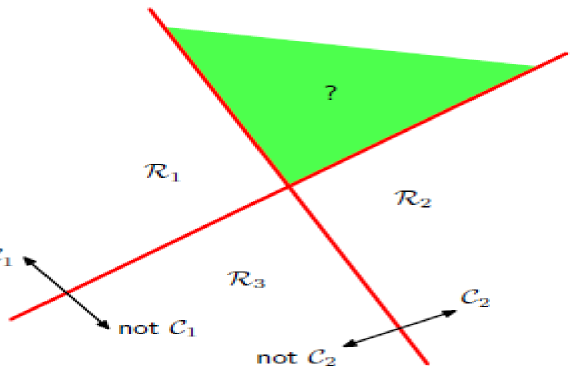


fig. 1.1.2 *3* class discriminant

Green region is ambiguous region because our discriminant is design to check point is in the class C_k or not in that class and hence class for this region remained undefined.

One vs k classifier:

Alternative approach for above problem is K(K-1)/2 discriminant functions, one for every possible pair of classes where K is total number of classes.

We will use K linear functions $Y_k(x) = W_k{}^T * X + W_{k0}$.

If $Y_k(x) > Y_j(x)$ then point x belongs to class C_k. The decision boundary between classes C_k *and* C_j is given by

$$(W_k - W_j)^T * X + (W_{k0} - W_{j0}) = 0$$

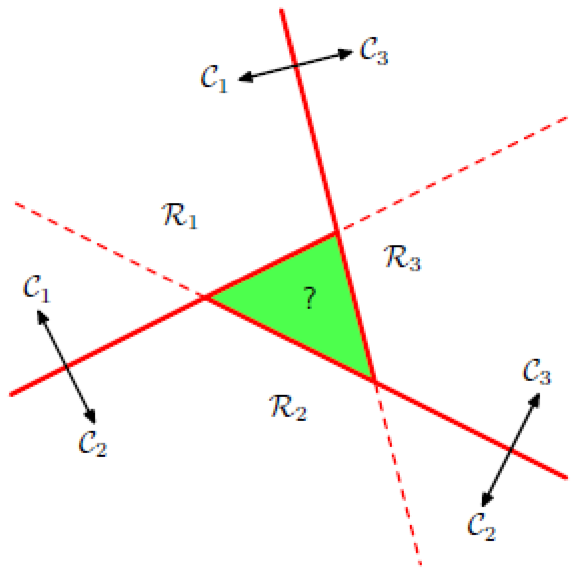


fig 1.1.3 3 class disriminator (1 vs 1)

If we compare fig 1.1.2 and fig 1.1.3, 3 class disriminator (1 vs 1) gives better classification 2 class classifier cause here we increases discrimination functions from K-1 to K(K-1)/2 which classifies input vector more accurately with less ambiguity.

1.2 Least Squares for classification:

If we define linear model as:

$y(x,w) = w_0 + w_1 x^1 + w_2 x^2 + \ldots\ldots + w_M x^M$

here M= Order of polynomial

So we already have result of points in t observations for each point on x axis. Here we tries to find $y(x,w)$ for corresponding x point. Ultimately we are learning W matrix from given observed points.

Once we get $y(x,w)$, we use least square error using our original observations t as follow:

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,w) - t_n\}^2 \qquad Eq\ 1.2.1$$

We try to minimize $E(w)$, so we are ultimately trying to get a curve for which summation projection of every point on the curve is minimum.

We will try to find W^* matrix from square error equation. We will get W^* from X and T matrices as :

$$\boldsymbol{W}^* = (\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\boldsymbol{T}}\,\boldsymbol{T} \qquad Eq\ 1.2.2$$

We will prove this in section 2.2.1 Mathematics later in this project report

Here for classification, we have to use 1 of K binary coding scheme which will convert class label discrete values to column of zero and one's in target vector T.

Least square approximates the conditional expectation E[**t**/**x**] of the target values given the input vector. For the binary coding scheme, this conditional expectation is given by the vector of posterior class probabilities[3]. however, these probabilities are approximated poorly, these approximations may have values outside the range (0, 1).

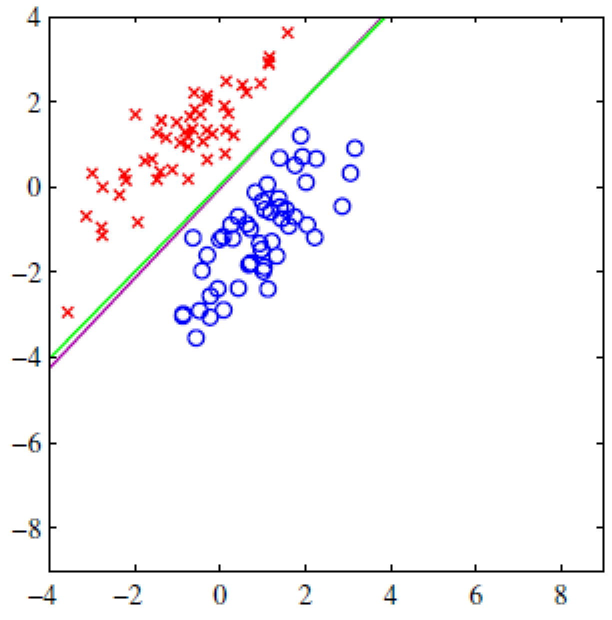


fig 1.2.1

Figure 1.2.1 shows response of linear classifier(purple line) separating two classes properly. But if we add more blue points away from existing decision boundary, it affects classifier and we get following result:

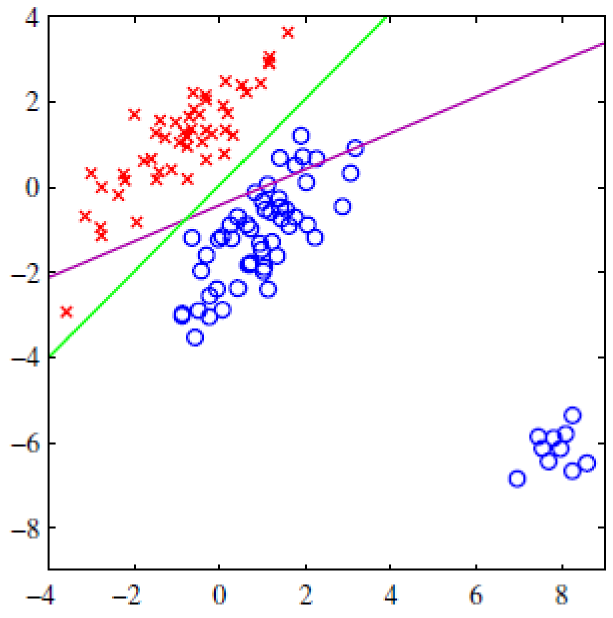


fig 1.2.2

Because of addition of new points, to reduce square error, classifier bent towards newly added points, misclassifying few original points. This is due inability of least square classifier's adapting changes and limited flexibility of linear models.

1.3 Fisher's linear discriminant:

With working in high dimensions is problematic because as dimension increases, requirement of number of features and examples increases exponentially and most of the time it leads to over-fitting. Data visualization and interpretation is difficult in high dimensions.

Fisher Linear Discriminant Analysis is method used in pattern recognition to find a linear combination of features which characterizes or separates two or more classes of objects or events by reducing the dimension of input vector[4].

In fisher discriminant analysis, we project high dimension space to minimum K-1 dimensions where K is number of classes[3]. This is importance of fisher analysis as it decides dimensionality according to number of classes, not number of features.

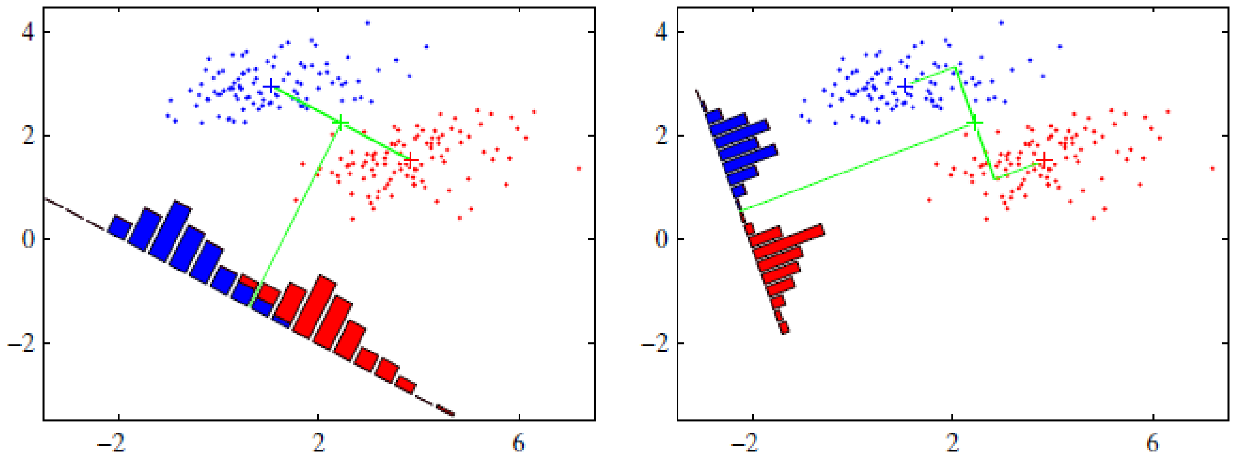


Fig 1.3.1

Fig 1.3.1 shows that using Fisher analysis we may solve the problem of dimensionality, but it leads to more serious problem, overlapping of classes in reduced dimension.

Being said this, our goal is to select projection which gives maximum separation between classes which could help us in classification.

To maximise class separation:

1. Maximizing inter-class (between-class) projected means
2. minimizing intra-class (within-class) variances

If m_1 and m_2 are respective means of two classes C1 and C2 & S_1^2 , S_2^2 are variance of C1 and C2, then total within class variance is given by $S_1^2 + S_2^2$.

We can formulate Fisher's Linear discriminator as :

$$J(w) = \frac{(m_1 - m_2)^2}{S_1^2 + S_2^2} \qquad \text{--------Eq 1.3.1}$$

we can generalise this equation for multiple classes:

Total within class variance = $\sum_{k=1}^{k} s_k^2$ we will denote it by matrix S_k

Total within class co-variance = S_w = $\sum_{k=1}^{k} S_k$

Mean of kth class = m_k

Total between class variance matrix given as

$$S_B = \sum_{k=1}^{k} N_k \ (m_k - m)(m_k - m)^T$$

Total mean of input matrix = m

Number of elements in kth class = N_k

Fisher's Linear discriminator finding ration is:

$$J(w) = \frac{W^T S_B W}{W^T S_w W}$$

If we maximise $J(w)$ for two classes, we will get following condition:

$$\boldsymbol{W \propto S_w^{-1} (m_1 - m_2)}$$

This Fisher's Linear Discriminator.

We can generalise this to multiple classes as:

maximise $\boldsymbol{J(w) = Transpose\,\{(W^T S_w W)^{-1} (W^T S_B W)\}}$

Eigen vectors of $S_w^{-1} S_B$ gives fisher parameters out of that first K-1 columns are dimensionality reduced parameters.

We are going to prove above result in section 2.2 Mathematics for Fisher Analysis

1.4 K- Nearest Neighbors:

Once we get fisher's discriminant parameters, we still need to classify reduced dimension features to classify them into corresponding classes. In this project, we are using K Nearest Neighbors(KNN) for output classification.

In KNN, we take distance between test point and all training point, we filter lowest 'K' distances, take vote of classes of these 'K' distances, the class which has more votes will be assigned to particular example[5].

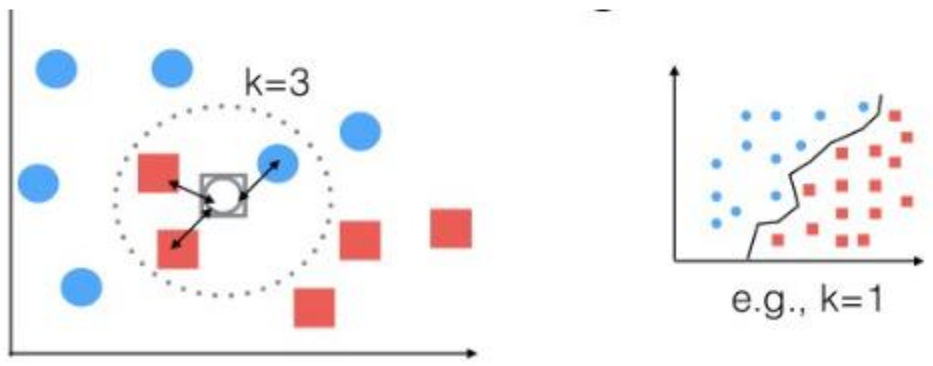


fig 1.4.1

Fig 1.4.1 shows KNN for K= 3 and K=1. White point is new and we want to classify it to either red class or blue class. For K= 3 we can see that red has higher number of votes, hence white point will be assigned to red class. K=1 is special case which helps to give boundary between two classes and it is useful for dataset having two classes. For more than two classes, k=1 is not accurate as it only looks for single nearest neighbor for classification[3] .

Value of K:

It is difficult to predict exact value of K which will give better classification without looking dataset. In general we have lot of examples with comparatively less classes, we can keep value of high value of K. But for this examples in dataset, high K value often leads to drastic decrease in accuracy.

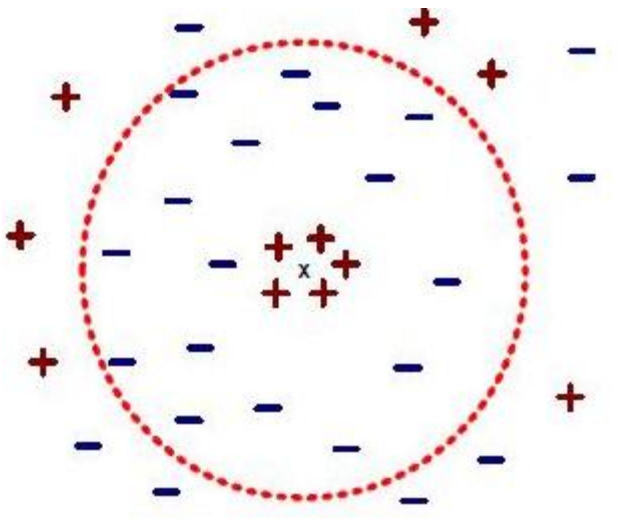


fig 1.4.2

In figure 1.4.2, if we keep K=5, we will get '+' class for new point which can be accurate result if look at visualization of data-points. But if we take K=15, it will classify the point in '-' class, which could be wrong classification. Best way to find K value is trial and test method for that particular dataset.

Distance Metric :

In this project, we are using L2 distance, which has the geometric interpretation of computing the Euclidean distance between two vectors[6].

$$d_{ij} = \sqrt{\sum(X_i - X_j)^2}$$

here, X_i = ith test point array of features, X_j = jth train point array of feature

We find d_{ij} for every j point in training set, sort d_{ij} in ascending order to get K nearest neighbors.

2. Approach

2.1 Datasets Used:

1. Wine dataset:
   Using chemical analysis to classify three wines from same region.
2. Wallpaper dataset:
   Find which of 17 wallpaper patterns contained in each image.
3. Taiji Dataset
   Find which of the 7 moves is starting from the MoCAP joint angles.

Dataset	Total Features	Total Observations		Total Classes
		Train	Test	
1) Wine	13	90	88	3
				{'0','1','2'}
2) Wallpaper	500	1700		17
				{P1, P2, PM, PG, CM, PMM, PGG, PMG, CMM, P4, P4M,P4G, P3, P3M1, P31M, P6, P6M }
3) Taiji	64	11361	3924	8
				{0, 1, 2, 3, 5,6,7,9}

2.2 Mathematics

2.2.1 Mathematics for Least Square Classifier:

Error function for square error can be given as :

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, w) - t_n\}^2$$

We need to minimize this equation, ie. we need to find W^* for which E(W^*) is minimum.

$$y(x, w) = w_0 + w_1 x^1 + w_2 x^2 + \dots\dots + w_M x^M \qquad here\ M = order$$

Lets convert polynomials into matrix form:

$$X = \begin{bmatrix} x_1^0 & x_1^1 & \dots\dots & x_1^M \\ x_2^0 & x_2^1 & \dots\dots & x_2^M \\ & & \dots\dots\dots & \\ x_N^0 & x_N^1 & \dots\dots & x_N^M \end{bmatrix}$$

here we need to understand that 1st column of X is column of 1's and other columns will get from dataset.

$$W = [w_0 \ w_1 \ w_2 \ \dots\dots\dots\dots\dots \ w_M]$$

$$T = [t_0 \ t_1 \ t_2 \ \dots\dots\dots\dots\dots \ t_N]$$

So error function can be represented as:

$E(W) = \frac{1}{2} \ (XW - T)^T \ (XW - T)$ -------------- > Eq 2.2.1.1

Taking transpose inside

$$E(W) = \frac{1}{2} \{ (W^T X^T - T^T)(XW - T)\}$$

inner multiplication,

$E(W) = \frac{1}{2} \{ W^T X^T X W - W^T X^T T - T^T X W + T^T T\}$ -------- > Eq 2.2.1.2

Differentiating w. r. t W

$$\frac{\partial\, E(W)}{\partial w} = \frac{1}{2}\left\{\frac{\partial\,(W^TX^TXW)}{\partial w} - \frac{\partial\,(W^TX^TT)}{\partial w} - \frac{\partial(T^TXW)}{\partial w} + \frac{\partial(T^TT)}{\partial w}\right\}$$

------- > Eq 2.2.1.3

Consider W^TX^TT

$$\begin{tikzcd} W^T \arrow[d] & X^T \arrow[d] & T \arrow[d] \\ 1\text{x}(M+1) & (M+1)\text{x}\,(N+1) & (N+1)\text{ x }1 \end{tikzcd}$$

$$\underbrace{1\text{x}(M+1) \quad \underbrace{(M+1)\text{x}\,(N+1) \quad (N+1)\text{ x }1}_{(M+1)\text{ x }1}}$$

(1 x 1) => Scalar

$$\frac{\partial\,(W^TX^TT)}{\partial w} = X^TT \quad and \quad \frac{\partial(T^TXW)}{\partial w} = \frac{\partial(W^TX^TT)^T}{\partial w} = X^TT$$

--------------Eq 2.2.1.4

Substituting Eq 2.2.1.4 in Eq 2.2.1.3 equating this equation to zero to get minimum :

$$\frac{\partial\, E(W)}{\partial w} = \frac{1}{2}\{\ 2X^TXW^* - X^TT - X^TT + 0\,\} = 0$$

$$2X^TXW^* - X^TT - X^TT = 0$$

$$2X^TXW^* = 2\,X^TT$$

$$X^TXW^* = X^TT$$

Multiplying $(X^TX)^{-1}$ on both sides,

$$(X^TX)^{-1}X^TXW^* = (X^TX)^{-1}X^TT$$

$$\boldsymbol{W^* = (X^TX)^{-1}X^TT}$$ **Eq 2.2.1.5**

Now minimized squared error function become :

$$\boldsymbol{E(w^*) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, w^*) - t_n\}^2}$$

where $Y(X, W^*) = XW^*$

1 to K target vector:

Here we need to understand that, for classification purpose, we have to convert target vector T into 1 to K matrix.

$$\begin{array}{cc} & \begin{array}{cccc} Y_k & Y_K & & \end{array} \\ \begin{bmatrix} 2 \\ 5 \\ 1 \\ 3 \end{bmatrix} = & \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

Suppose class label for particular example m is 3 then it 1:k encoding is given by

[0; 0; 1; 0; 0] ie. 3rd row of mth column is 1 other elements in that column are 0.

So out T vector have size (m+1)x K where K is total number of classes. This don't hurt our $W^* = (X^T X)^{-1} X^T T$ because while classification we ultimately choose max value of $Y(X, W^*)$ out of K possible values and predict the label on basis of that max value from class categories. In different words, anyhow we discards K-1 values of $Y(X, W^*)$ for every feature vector point in X.

2.2.2 Mathematics for Fisher Discriminant analysis:

We will try to generalise 2 class fisher equations for multiple classes with some modifications.

step 1: Convert Fisher analysis equations into matrix form

If m_1 and m_2 are respective means of two classes C1 and C2 & S_1^2 , S_2^2 are variance of C1 and C2, then total within class variance is given by $S_1^2 + S_2^2$.

We can formulate Fisher's Linear discriminator as :

$$J(w) = \frac{(m_1 - m_2)^2}{S_1^2 + S_2^2} \qquad \text{---------------Eq 2.2.2.1}$$

we can write m1 and m2 in matrix form as:

$$m_{01} = \frac{\sum_{n=1}^{N1} X_n}{N1} \quad \& \quad m_{02} = \frac{\sum_{n=1}^{N2} X_n}{N2}$$

$$(m_1 - m_2)^2 = (W^T m_{01} - W^T m_{02})^2$$

$$= Tr[(W^T m_{01} - W^T m_{02})^T (W^T m_{01} - W^T m_{02})]$$

Taking transpose inside

$$= Tr[(W^T (m_{01} - m_{02}))^T W^T (m_{01} - m_{02})]$$

Using Trans[AB] = Trans[B] Trans[A] property

$$= Tr[(m_{01} - m_{02})^T W \; W^T (m_{01} - m_{02})]$$

Using Trans[ABCD] = Trans[BCDA] = Trans[CDAB] property

$$= Tr[W^T (m_{01} - m_{02}) (m_{01} - m_{02})^T W]$$

$$(\boldsymbol{m_1} - \boldsymbol{m_2})^2 = \boldsymbol{W^T S_B W} \qquad \text{---------------Eq 2.2.2.2}$$

here

$$S_B = (m_{01} - m_{02}) (m_{01} - m_{02})^T$$

$$m_{01} = \frac{\sum_{n=1}^{N1} X_n}{N1} \quad \& \quad m_{02} = \frac{\sum_{n=1}^{N2} X_n}{N2}$$

$$N1\, m_{01} = \sum_{n=1}^{N1} X_n \quad \& \quad N2\, m_{02} = \sum_{n=1}^{N2} X_n$$

$$\sum_{n=1}^{N1} X_n + \sum_{n=1}^{N2} X_n = \sum_{n=1}^{N} X_n = N\ m = N1\, m_{01} + N2\, m_{02}$$

N1 + N2 = N = Total data points in input vector

$N_k = Data\ points\ in\ Kth\ class$

m = mean of input vector,

now S_B can generalise as

$$\boldsymbol{S_B = \sum_{k=1}^{k} N_k\ (m_k - m)(m_k - m)^T}$$ --------Eq 2.2.2.3

Let's find between class co-variance matrix:

$$S_k^2 = \sum_{n\, \in\, C_k} (Y_k(x) - m_{0k})^2 = \sum(W^T X_n - W^T m_{0k})^2$$

consider $(W^T X_n - W^T m_{0k}) =$

$$= Tr[(W^T X_n - W^T m_{0k})^T (W^T x_n - W^T m_{0k})]$$

Taking transpose inside

$$= Tr[(W^T (X_n - m_{0k}))^T W^T (X_n - m_{0k})]$$

Using Trans[AB] = Trans[B] Trans[A] property

$$= Tr[(X_n - m_{0k})^T W\ W^T (X_n - m_{0k})\ ]$$

Using Trans[ABCD] = Trans[BCDA] = Trans[CDAB] property

$$= Tr[\, W^T (X_n - m_{0k})\ (X_n - m_{0k})^T\ W\ ]$$

$$S_k^2 = \sum W^T \; (X_n - \; m_{0k}) \; (X_n - \; m_{0k})^T \; W$$

$$S_k^2 = \; W^T \sum (X_n - \; m_{0k}) \; (X_n - \; m_{0k})^T \; W$$

$$S_w \; = \; (X_n - \; m_{0k}) \; (X_n - \; m_{0k})^T$$

$$\boldsymbol{S_k^2 = \; W^T \, S_w \, W} \qquad \text{-----------}\rightarrow \text{ Eq 2.2.2.4}$$

This is our within class covariance matrix.

Substituting Eq2.2.2.2 and Eq2.2.2.4 in Eq2.2.2.1

$$\boldsymbol{J(W) = \frac{W^T \, S_B \, W}{W^T \, S_W \, W}} \qquad \text{------------- > \quad Eq 2.2.2.5}$$

<u>Step 2</u>: Maximise J(W):

$$\frac{\partial \, J(W)}{\partial W} = \frac{\partial \frac{W^T \, S_B \, W}{W^T \, S_W \, W}}{\partial W} = 0$$

$$\frac{\partial \, J(W)}{\partial W} = \; (W^T \, S_W \, W) \frac{\partial \, (\, W^T \, S_B \, W)}{\partial W} - \; (W^T \, S_B \, W) \frac{\partial \, (\, W^T \, S_W \, W)}{\partial W} = 0$$

$$2\underbrace{(W^T \, S_W \, W)}_{\text{Scalar}} \, S_B \, W \; = 2 \; \underbrace{(W^T \, S_B \, W)}_{\text{scalar as size } W^T \text{ =1xD and size W = D x1}} S_W \, W$$

$$\boldsymbol{S_B \, W \; = \; \lambda \, S_W \, W} \qquad \text{-----------} \; (\lambda \; represents \; scalar \; vlaue)$$

$$\boldsymbol{\lambda W \; \propto S_W{}^{-1} \, S_B} \qquad \text{-- --------}\rightarrow \text{ Eq 2.2.2.6}$$

We select first K-1 columns of eigen vector of $S_W{}^{-1} S_B$ which gives fisher's parameters which we multiply with input vector to get K-1 dimensions.

here matrix $S_k = \sum_{\boldsymbol{n} \,\epsilon\, \boldsymbol{C_k}} \left((X_n - m_k)\,(X_n - m_k)^T \right)$

& $S_W = \sum_{k=1}^{k} S_k$

his equation we have used in out project to find fisher discriminant. The weight values are determined by those eigenvectors of $S_W{}^{-1} S_B$.

Note:

S_B is compose of K matrices which is outer product of two vectors with rank 1. Only K-1 of these matrices are independent of result constraint . So there are at most K-1 non zero eigenvalues[4]. Hence we can project to K-1 dimension without changing inherent properties of J(W).

2.2.3 Mathematics for K Nearest Neighbor:

As discussed earlier, we are using Euclidean distance between test point and all train points, sort minimum K distances, compare class of K lowest distance points and select maximum frequent class.

$$d_{ij} = \sqrt{\sum (X_i - X_j)^2}$$

here, X_i = ith test point array of features, X_j = jth train point array of feature

We find d_{ij} for every j point in training set, sort d_{ij} in ascending order to get K nearest neighbors.

2.3 Confusion and Classification Matrix:

For given test vector $x^1,\ x^2 \ldots.., x^M$ and test labels $Y^1,\ Y^2 \ldots.., Y^M$ we can interpret classification in terms of classified label verses true label for particular example by using confusion matrix[7]. Size of confusion matrix is K by K where K is total number of classes.

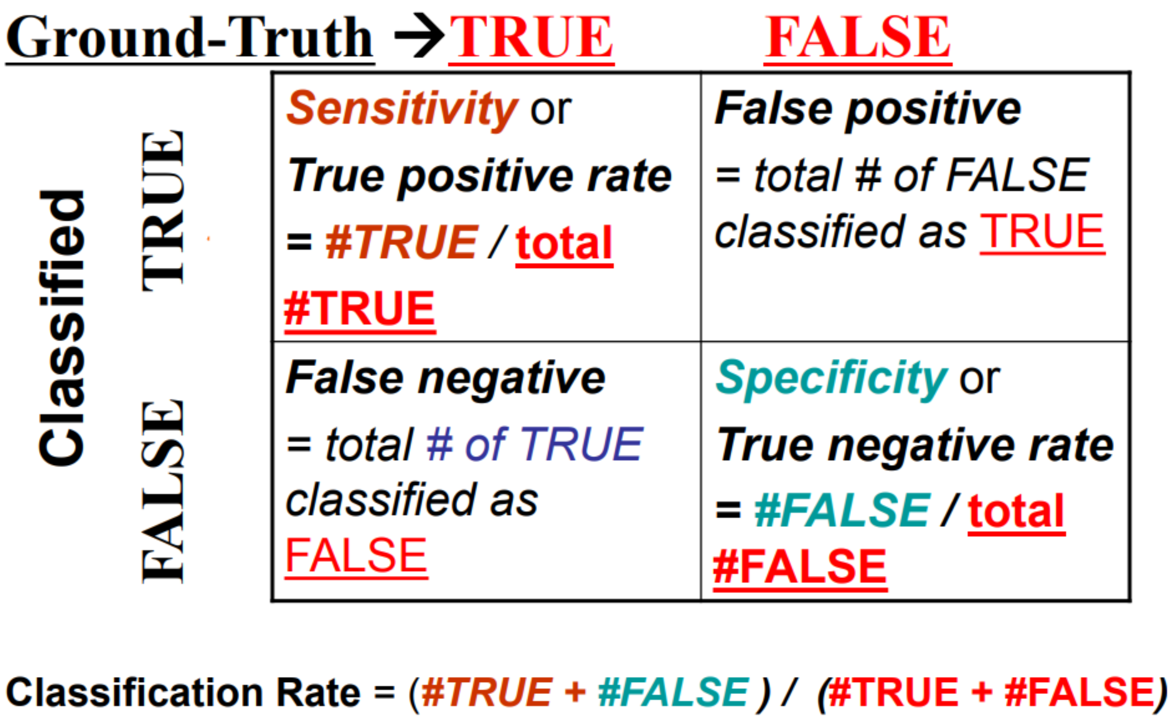
Let's consider example of 2 classes +1 and -1.

	Class +1 (True)	**Class -1 (True)**	**Total**
Class +1 (Classified)	Ntp	Nfn	**Np**
Class -1 (Classified)	Nfp	Ntn	**Nn**
Total	**Ntp + Nfp**	**Nfn + Ntn**	**N**

Error rate = (Nfp + Nfn)/N

true positive (or recall) rate =Ntp/Np

false positive rate (or false alarm rate) = Nfp/Nn



Classified ↓ / **Ground-Truth** →	**TRUE**	**FALSE**
TRUE	***Sensitivity*** or ***True positive rate*** = ***#TRUE*** / **total #TRUE**	***False positive*** = *total # of FALSE classified as TRUE*
FALSE	***False negative*** = *total # of TRUE classified as FALSE*	***Specificity*** or ***True negative rate*** = ***#FALSE*** / **total #FALSE**

Classification Rate = (*#TRUE + #FALSE*) / (*#TRUE + #FALSE*)

fig 2.3.1 Confusion/Classification (Lecture notes)

2.4 Matlab functions

Following table contains custom Matlab files we used in code to show our result:

Function/variable	Use
1) Start.m	Main matlab file. Code needs to be executed from this file
2) TrainingModel.m	Learns weight parameters for least square classifier
3) Prediction.m	Assign class labels to observations using least square model
4) visualizeBoundaries.m	Find boundaries for classes of wine dataset
5) fisherDiscriminator_X.m	Reduce dimensions using Fisher's analysis
6) KNN Classifier.m	Assign class labels to reduced dimension vector using KNN classification

3. Results

3.1 Least Square Classifier:

a) Accuracy and Standard Deviation :

	Wine	Wallpaper	Taiji
Train Accuracy (%)	100	96.76	96.13
Test Accuracy (%)	97.14	61.71	92.87
Train std	0	0.0329	0.1094
Test std	0. 0495	0.2696	0.2017

Table 3.1.1

1. Accuracy of wine training data set is 100% ! From test confusion matrix (Table 3.1.3) we can say that 3 points are mislabelled which costs accuracy reduced by approx 3%.

2. Accuracy of Wallpaper test dataset is poorest amongst all dataset on least square classifier. Although training accuracy is more than 95%, test accuracy is around 61% which states high variance. We need to notice that drastic increase in standard deviation of test data as compare to that of train data. This may caused due to more features , classes and comparatively less observations. By increasing observations, this variance can be minimized.

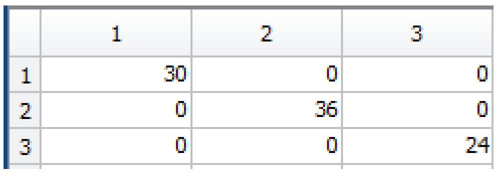
3. Taiji data set is suited for least square classifier. Moderate features, classes and comparatively more observation data controls variance in this dataset as both train and test accuracies are more than 90%

b) Confusion and Classification Matrix:

Wine :

Confusion Matrix

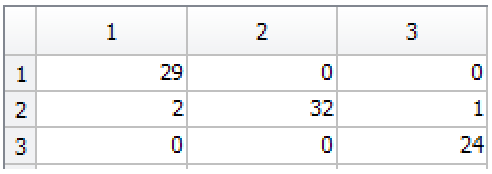
Train



	1	2	3
1	30	0	0
2	0	36	0
3	0	0	24

Table 3.1.2

Test

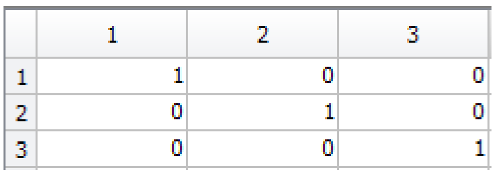


	1	2	3
1	29	0	0
2	2	32	1
3	0	0	24

Table 3.1.3

Classification Matrix:

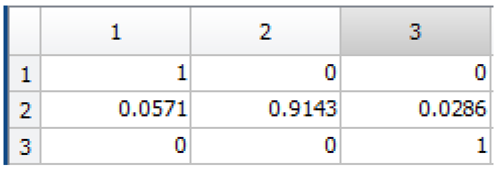
Train



	1	2	3
1	1	0	0
2	0	1	0
3	0	0	1

Table 3.1.4

Test



	1	2	3
1	1	0	0
2	0.0571	0.9143	0.0286
3	0	0	1

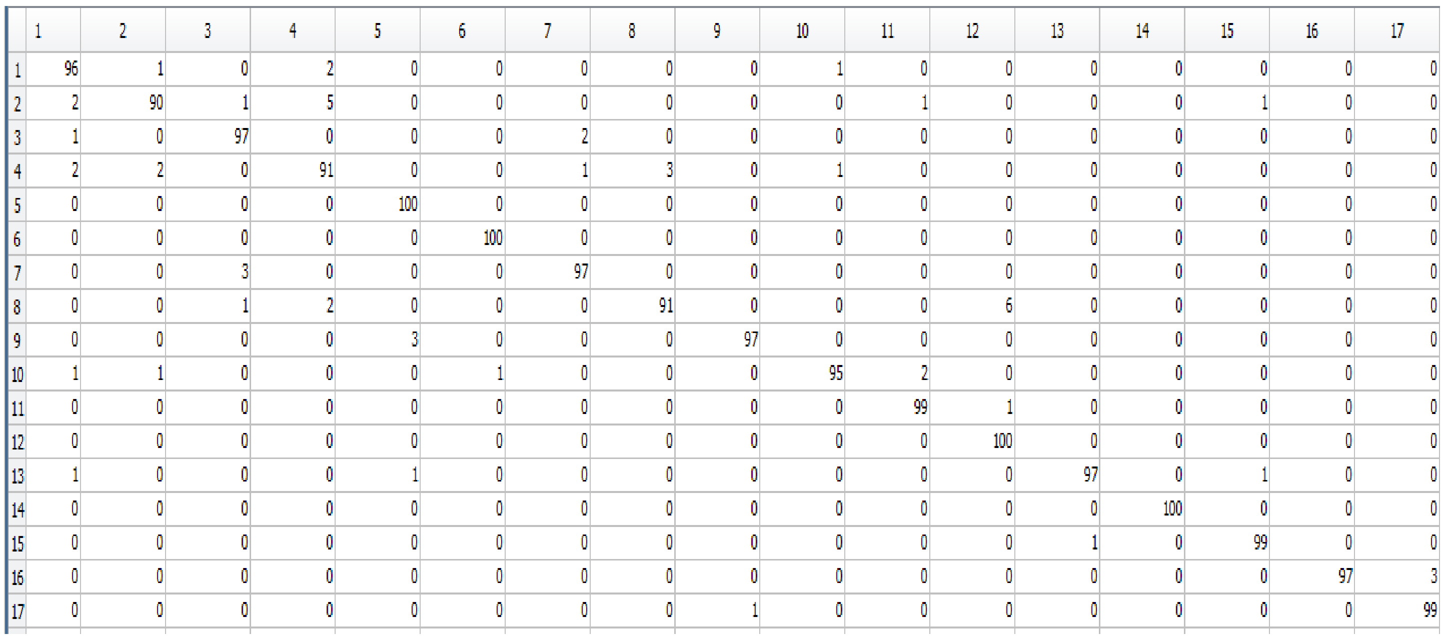
Table 3.1.5

1. Train confusion matrix (Table 3.1.2) shows that all points are classified as true labels.
2. On other hand, test confusion matrix (Table 3.1.3) shows that 2 observations of class 1 true label classified as class 2. This is advantage of confusion matrix as we can statistically compare result of train and test observations.
3. All rows of classification matrix sums to one shows that our confusion matrix representation is proper.

Wallpaper:

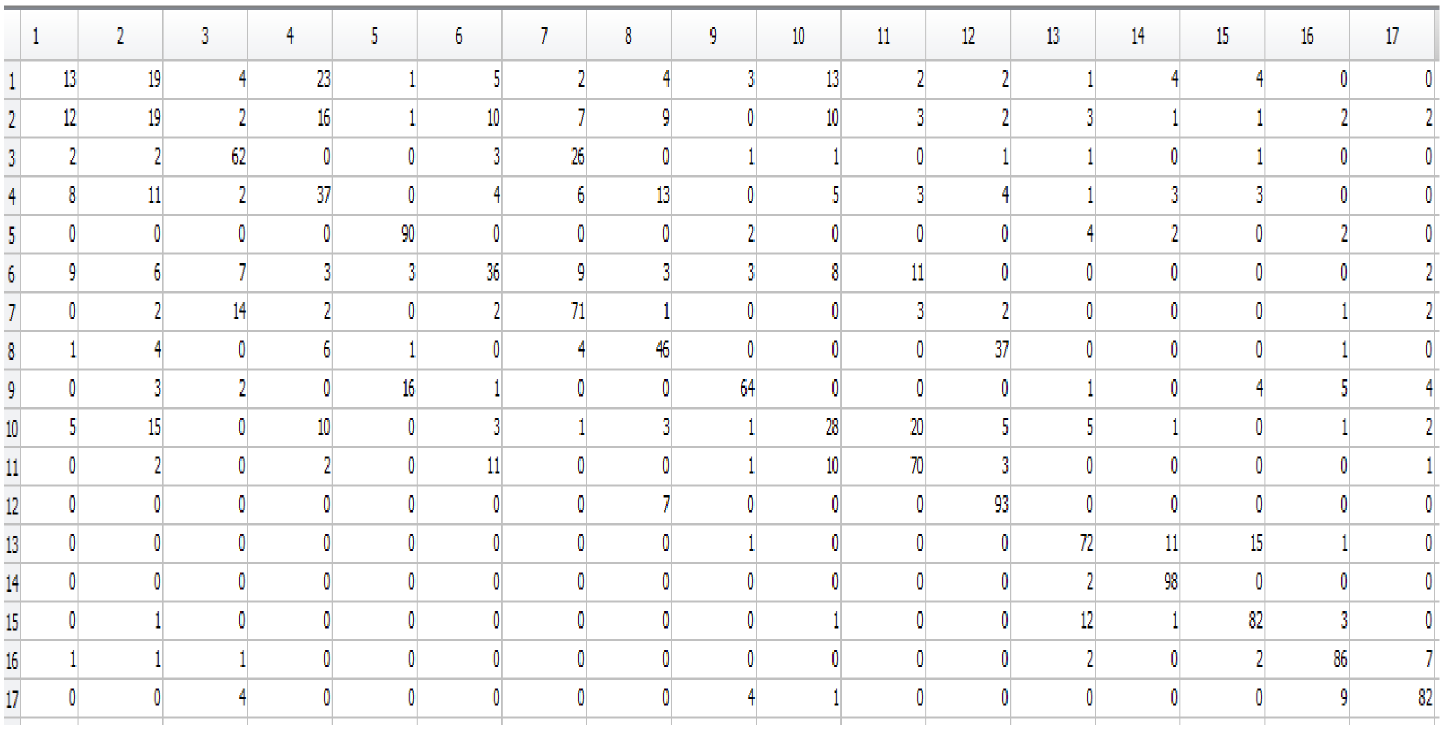
Confusion Matrix

Train: Table 3.1.6



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	96	1	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0
2	2	90	1	5	0	0	0	0	0	0	1	0	0	0	1	0	0
3	1	0	97	0	0	0	2	0	0	0	0	0	0	0	0	0	0
4	2	2	0	91	0	0	1	3	0	1	0	0	0	0	0	0	0
5	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0
7	0	0	3	0	0	0	97	0	0	0	0	0	0	0	0	0	0
8	0	0	1	2	0	0	0	91	0	0	0	6	0	0	0	0	0
9	0	0	0	0	3	0	0	0	97	0	0	0	0	0	0	0	0
10	1	1	0	0	0	1	0	0	0	95	2	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	99	1	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0
13	1	0	0	0	1	0	0	0	0	0	0	0	97	0	1	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	1	0	99	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	97	3
17	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	99

Test : Table 3.1.7

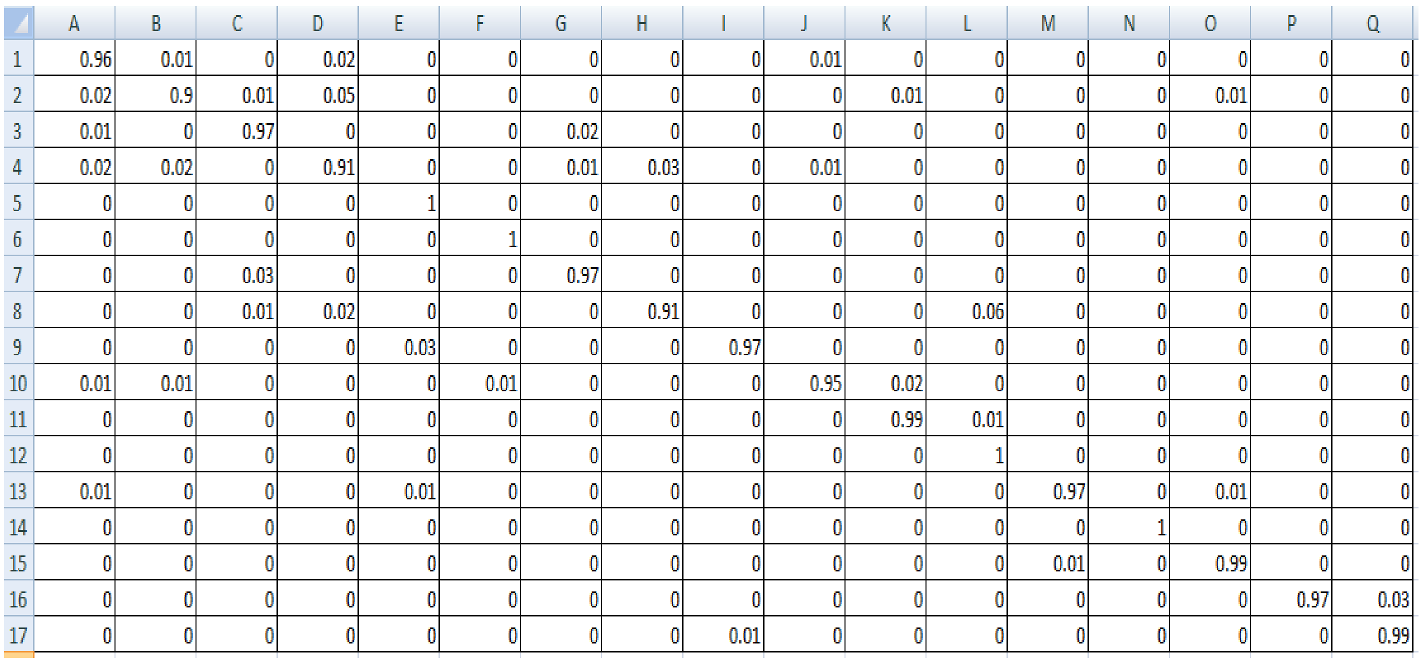


	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	13	19	4	23	1	5	2	4	3	13	2	2	1	4	4	0	0
2	12	19	2	16	1	10	7	9	0	10	3	2	3	1	1	2	2
3	2	2	62	0	0	3	26	0	1	1	0	1	1	0	1	0	0
4	8	11	2	37	0	4	6	13	0	5	3	4	1	3	3	0	0
5	0	0	0	0	90	0	0	0	2	0	0	0	4	2	0	2	0
6	9	6	7	3	3	36	9	3	3	8	11	0	0	0	0	0	2
7	0	2	14	2	0	2	71	1	0	0	3	2	0	0	0	1	2
8	1	4	0	6	1	0	4	46	0	0	0	37	0	0	0	1	0
9	0	3	2	0	16	1	0	0	64	0	0	0	1	0	4	5	4
10	5	15	0	10	0	3	1	3	1	28	20	5	5	1	0	1	2
11	0	2	0	2	0	11	0	0	1	10	70	3	0	0	0	0	1
12	0	0	0	0	0	0	0	7	0	0	0	93	0	0	0	0	0
13	0	0	0	0	0	0	0	0	1	0	0	0	72	11	15	1	0
14	0	0	0	0	0	0	0	0	0	0	0	0	2	98	0	0	0
15	0	1	0	0	0	0	0	0	0	1	0	0	12	1	82	3	0
16	1	1	1	0	0	0	0	0	0	0	0	0	2	0	2	86	7
17	0	0	4	0	0	0	0	0	4	1	0	0	0	0	0	9	82

Wallaper

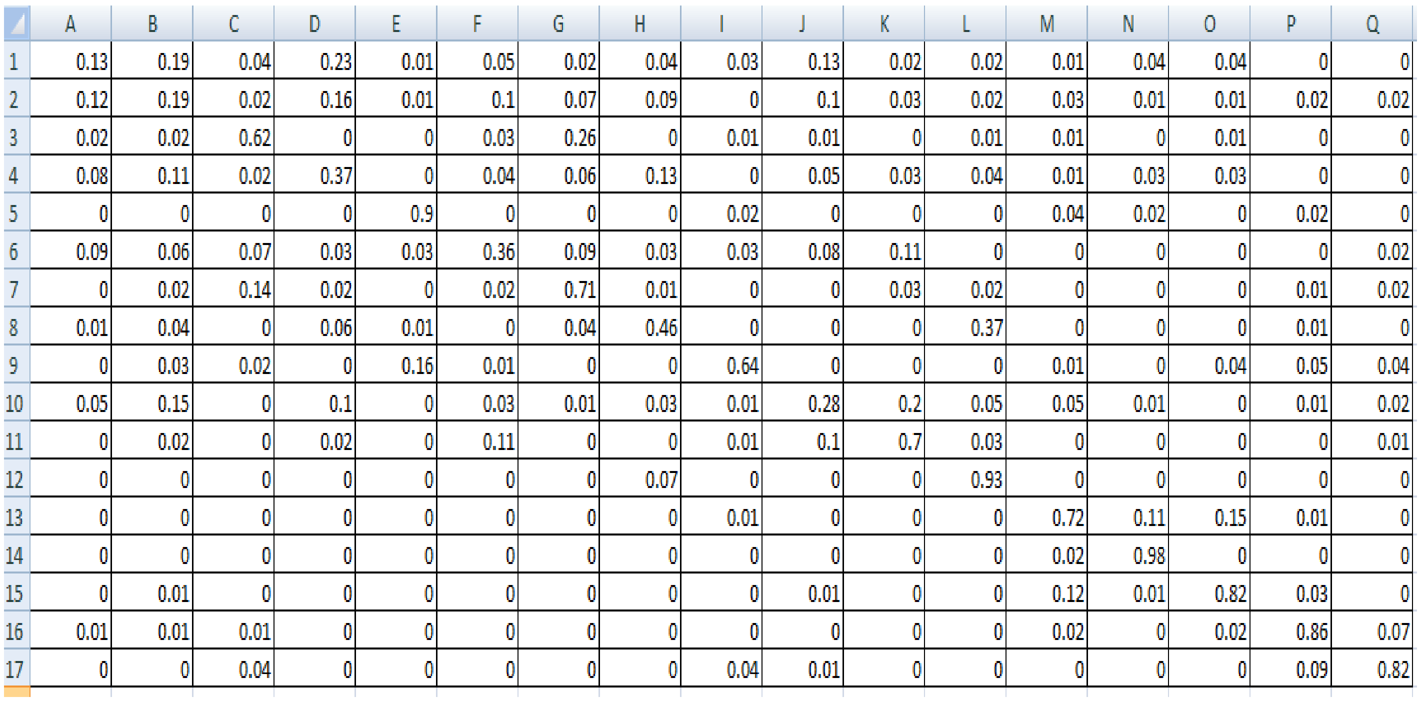
Classification Matrix:

Train: Table 3.1.8



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	0.96	0.01	0	0.02	0	0	0	0	0	0.01	0	0	0	0	0	0	0
2	0.02	0.9	0.01	0.05	0	0	0	0	0	0	0.01	0	0	0	0.01	0	0
3	0.01	0	0.97	0	0	0	0.02	0	0	0	0	0	0	0	0	0	0
4	0.02	0.02	0	0.91	0	0	0.01	0.03	0	0.01	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0.03	0	0	0	0.97	0	0	0	0	0	0	0	0	0	0
8	0	0	0.01	0.02	0	0	0	0.91	0	0	0	0.06	0	0	0	0	0
9	0	0	0	0	0.03	0	0	0	0.97	0	0	0	0	0	0	0	0
10	0.01	0.01	0	0	0	0.01	0	0	0	0.95	0.02	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0.99	0.01	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
13	0.01	0	0	0	0.01	0	0	0	0	0	0	0	0.97	0	0.01	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0.01	0	0.99	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.97	0.03
17	0	0	0	0	0	0	0	0	0.01	0	0	0	0	0	0	0	0.99

Test: Table 3.1.9



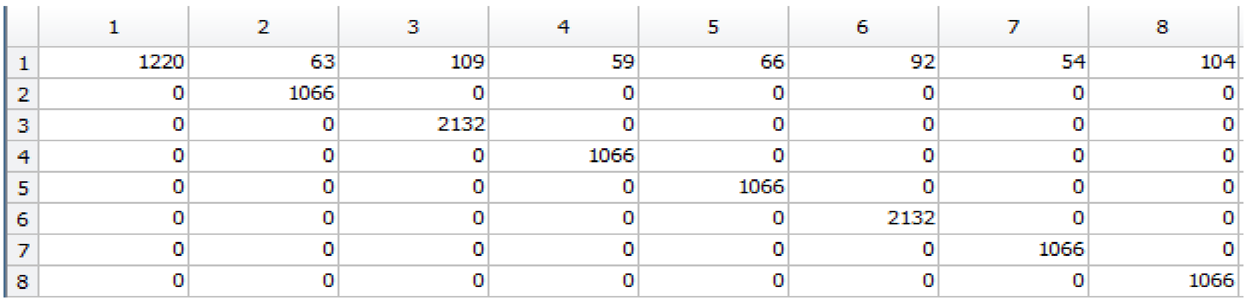
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	0.13	0.19	0.04	0.23	0.01	0.05	0.02	0.04	0.03	0.13	0.02	0.02	0.01	0.04	0.04	0	0
2	0.12	0.19	0.02	0.16	0.01	0.1	0.07	0.09	0	0.1	0.03	0.02	0.03	0.01	0.01	0.02	0.02
3	0.02	0.02	0.62	0	0	0.03	0.26	0	0.01	0.01	0	0.01	0.01	0	0.01	0	0
4	0.08	0.11	0.02	0.37	0	0.04	0.06	0.13	0	0.05	0.03	0.04	0.01	0.03	0.03	0	0
5	0	0	0	0	0.9	0	0	0	0.02	0	0	0	0.04	0.02	0	0.02	0
6	0.09	0.06	0.07	0.03	0.03	0.36	0.09	0.03	0.03	0.08	0.11	0	0	0	0	0	0.02
7	0	0.02	0.14	0.02	0	0.02	0.71	0.01	0	0	0.03	0.02	0	0	0	0.01	0.02
8	0.01	0.04	0	0.06	0.01	0	0.04	0.46	0	0	0	0.37	0	0	0	0.01	0
9	0	0.03	0.02	0	0.16	0.01	0	0	0.64	0	0	0	0.01	0	0.04	0.05	0.04
10	0.05	0.15	0	0.1	0	0.03	0.01	0.03	0.01	0.28	0.2	0.05	0.05	0.01	0	0.01	0.02
11	0	0.02	0	0.02	0	0.11	0	0	0.01	0.1	0.7	0.03	0	0	0	0	0.01
12	0	0	0	0	0	0	0	0.07	0	0	0	0.93	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0.01	0	0	0	0.72	0.11	0.15	0.01	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0.02	0.98	0	0	0
15	0	0.01	0	0	0	0	0	0	0	0.01	0	0	0.12	0.01	0.82	0.03	0
16	0.01	0.01	0.01	0	0	0	0	0	0	0	0	0	0.02	0	0.02	0.86	0.07
17	0	0	0.04	0	0	0	0	0	0.04	0.01	0	0	0	0	0	0.09	0.82

Rather than maximum value to diagonal elements, wallpaper test confusion matrix (Table 3.1.6)shows number of observations misclassified. This itself shows low accuracy for wallpaper dataset on least square classifier.

Taiji

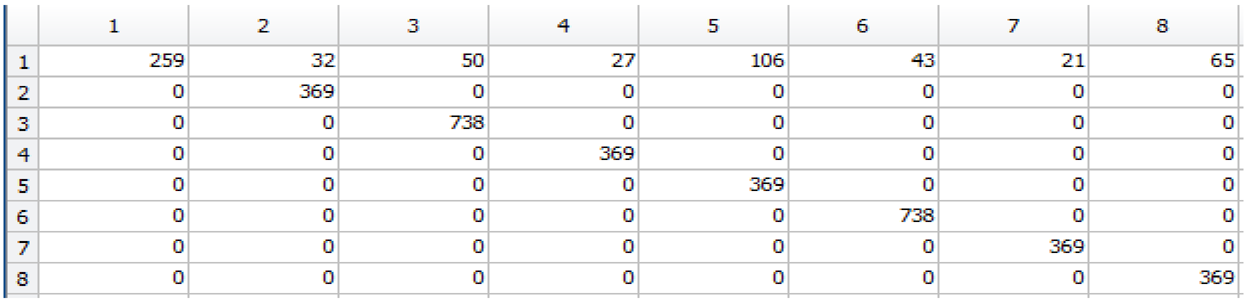
Confusion Matrix:

Train: Table 3.1.10



	1	2	3	4	5	6	7	8
1	1220	63	109	59	66	92	54	104
2	0	1066	0	0	0	0	0	0
3	0	0	2132	0	0	0	0	0
4	0	0	0	1066	0	0	0	0
5	0	0	0	0	1066	0	0	0
6	0	0	0	0	0	2132	0	0
7	0	0	0	0	0	0	1066	0
8	0	0	0	0	0	0	0	1066

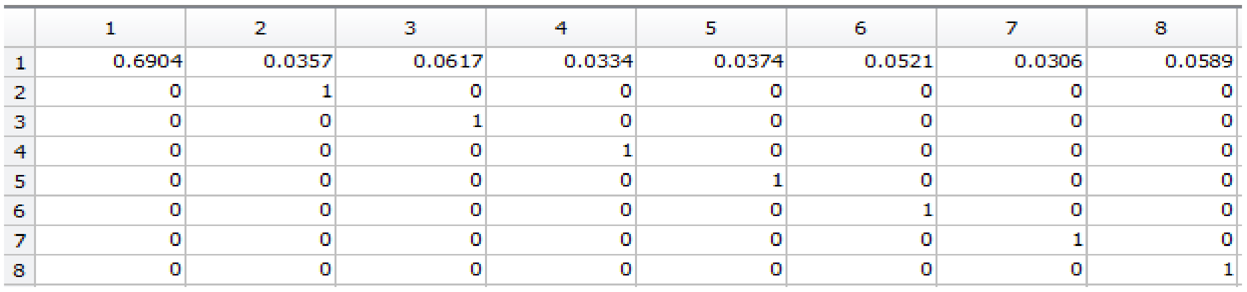
Test: Table 3.1.11



	1	2	3	4	5	6	7	8
1	259	32	50	27	106	43	21	65
2	0	369	0	0	0	0	0	0
3	0	0	738	0	0	0	0	0
4	0	0	0	369	0	0	0	0
5	0	0	0	0	369	0	0	0
6	0	0	0	0	0	738	0	0
7	0	0	0	0	0	0	369	0
8	0	0	0	0	0	0	0	369

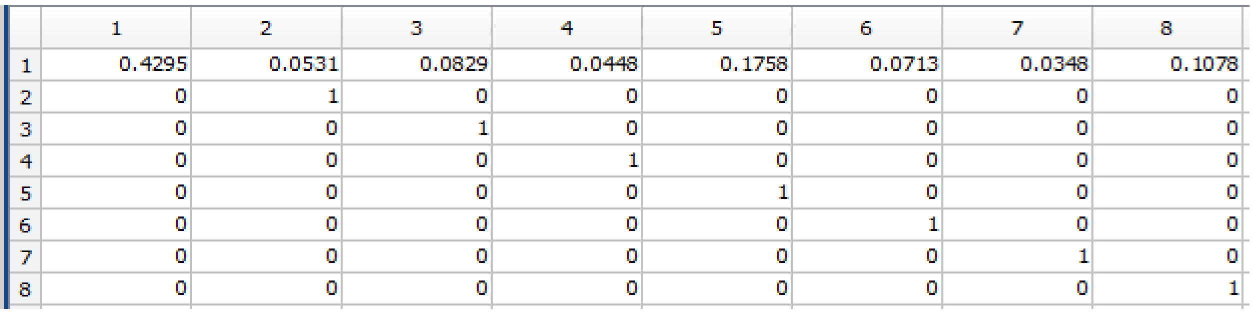
Classification matrix:

Train: Table 3.1.12



	1	2	3	4	5	6	7	8
1	0.6904	0.0357	0.0617	0.0334	0.0374	0.0521	0.0306	0.0589
2	0	1	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0
4	0	0	0	1	0	0	0	0
5	0	0	0	0	1	0	0	0
6	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	1

Test: Table 3.1.13



	1	2	3	4	5	6	7	8
1	0.4295	0.0531	0.0829	0.0448	0.1758	0.0713	0.0348	0.1078
2	0	1	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0
4	0	0	0	1	0	0	0	0
5	0	0	0	0	1	0	0	0
6	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	1

The 1st row show that from every class labels, least square classified these observations to Class '0'. Otherwise the performance is perfect for other classes.

C) Classification diagram:

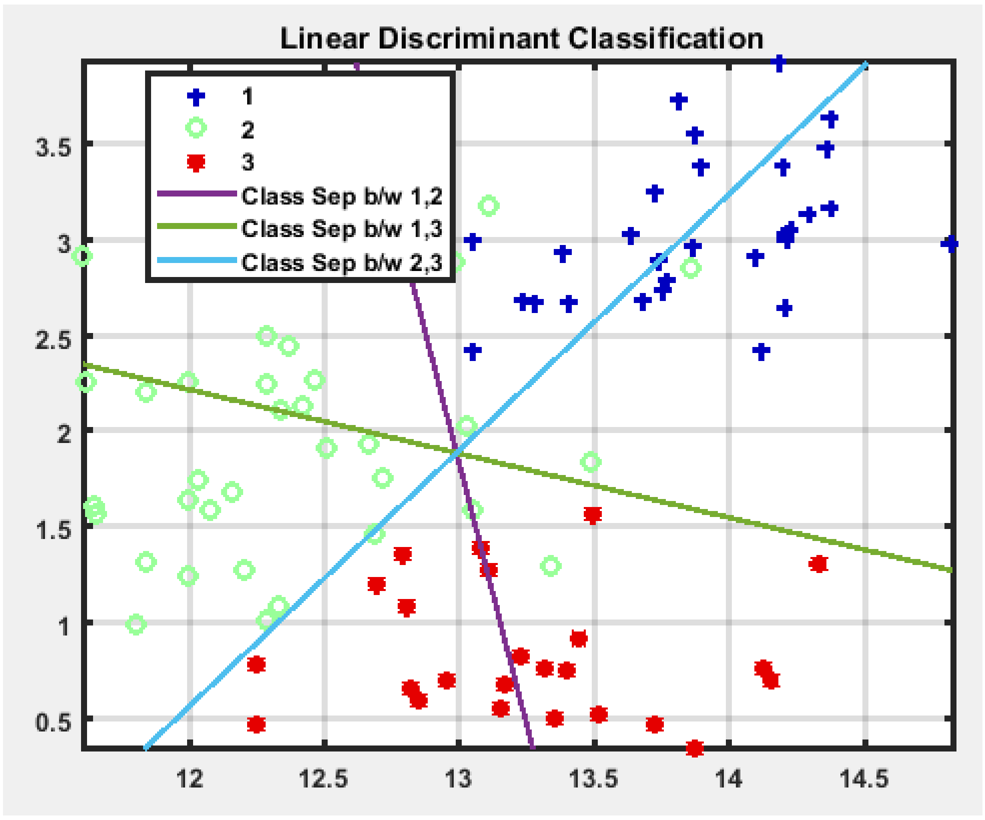


fig 3.1.1

Fig 3.1.1 show linear boundaries between 3 classes of wine dataset. For visualization purpose we took only two feature out of 13 available features from dataset. As total classes are 3, it was possible to reduce dimension to 2 (K-1, k= total number of classes).

3.2 Fisher Discriminant Analysis:

a) Accuracy and Standard Deviation for K=5

	Wine	Wallpaper	Taiji
Train Accuracy (%)	98.89	97	98.27
Test Accuracy (%)	96	64.42	87.73
Train std	0.0192	0.0294	0.0488
Test std	0.0431	0.2486	0.2104

Table 3.2.1

b) Accuracy and Standard Deviation for K=9

	Wine	Wallpaper	Taiji
Train Accuracy (%)	98.89	97	98.03
Test Accuracy (%)	98.10	68.59	89.35
Train std	0.0192	0.0294	0.0558
Test std	0.0330	0.2166	0.1919

Table 3.2.2

C) Accuracy and Standard Deviation for K=15

	Wine	Wallpaper	Taiji
Train Accuracy (%)	98.89	96.41	97.81
Test Accuracy (%)	99.05	67.83	90.46
Train std	0.0192	0.0400	0.0620
Test std	0.0165	0.2507	0.1691

Table 3.2.3

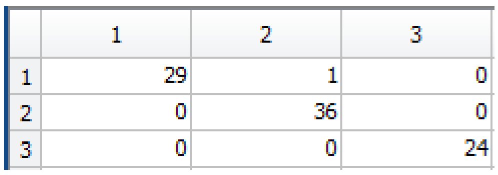
1. If you compare least square accuracy and standard deviation table(Table 3.1.1) with corresponding Fisher discriminator table(3.2.2), standard deviation is reduced. Our main criteria for using Fisher's discriminator - reduce within class variance is satisfied.
2. Test accuracy of fisher + KNN classifier is slightly increased on test set of every dataset.
3. With increase in value of K, test accuracy increased where as train accuracy and standard deviation remained constant(Table 3.2.1 and Table 3.2.2)
4. Wallpaper datasets have less number of training examples as compare to Taiji, so at K = 15, wallpaper test accuracy decreased where Taiji test accuracy still increased. This is because wallpaper has low observations and more classes as compare to Taiji dataset.
5. As like least square classifier, fisher + KNN also has high variance on wallpaper dataset. Being said that, we have to note that, wallpaper dataset has 500 features and least square classifier utilizes all of these features. On contrary, after fisher's projection, number of dimensions converted into 16, still fisher + KNN gives better performance than least square.

d) Confusion and classification Matrix for **K =9**

Wine :

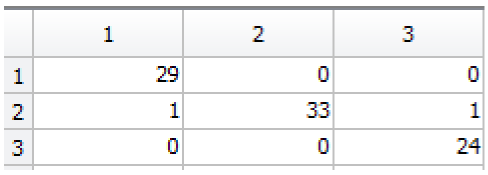
Confusion Matrix:

Train Table 3.2.4



	1	2	3
1	29	1	0
2	0	36	0
3	0	0	24

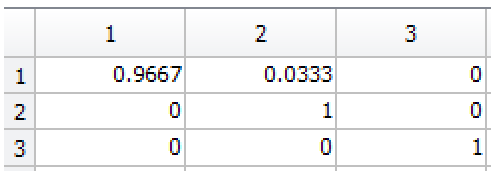
Test Table 3.2.5



	1	2	3
1	29	0	0
2	1	33	1
3	0	0	24

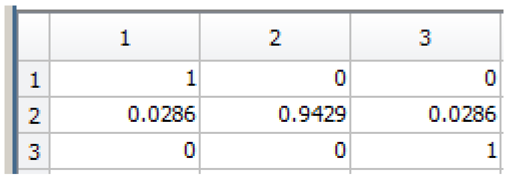
Classification Matrix:

Train Table 3.2.6



	1	2	3
1	0.9667	0.0333	0
2	0	1	0
3	0	0	1

Test Table 3.2.7

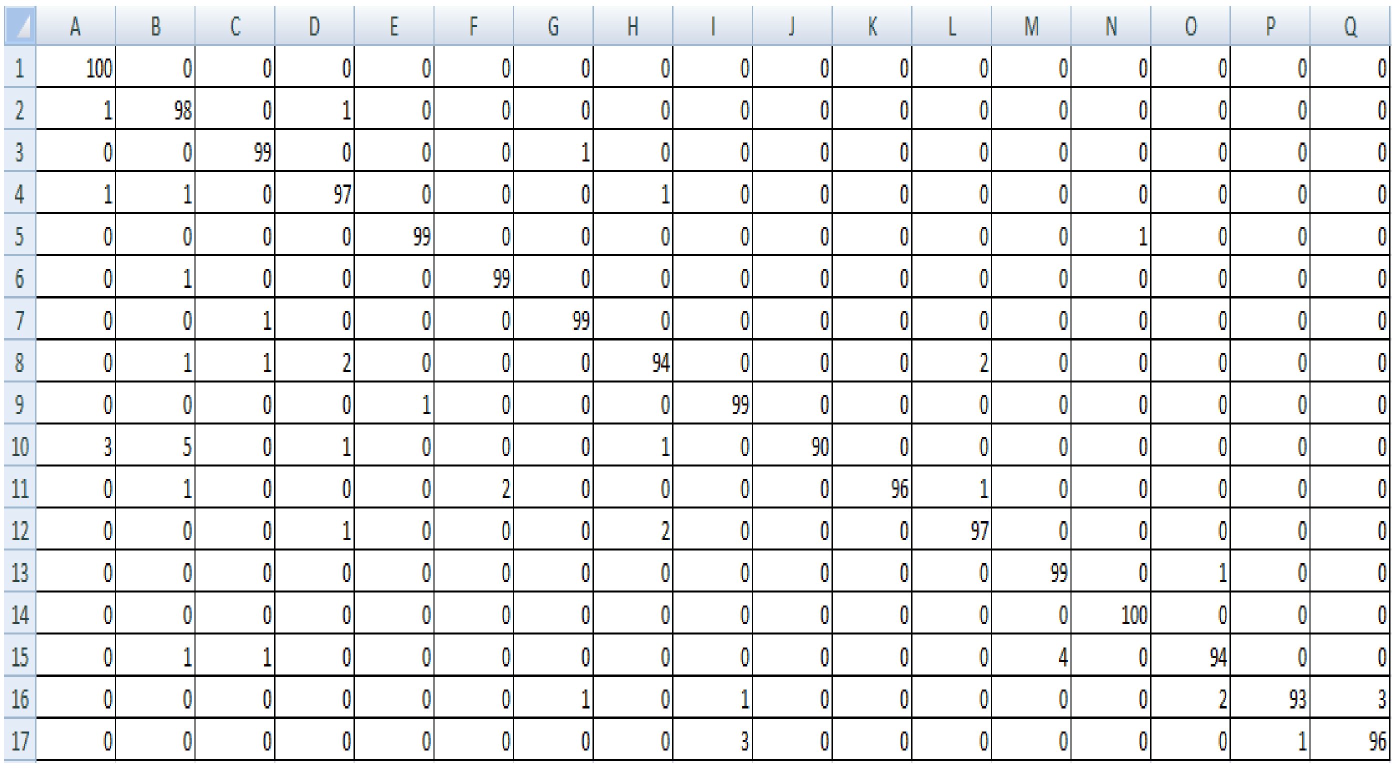


	1	2	3
1	1	0	0
2	0.0286	0.9429	0.0286
3	0	0	1

Wallpaper:

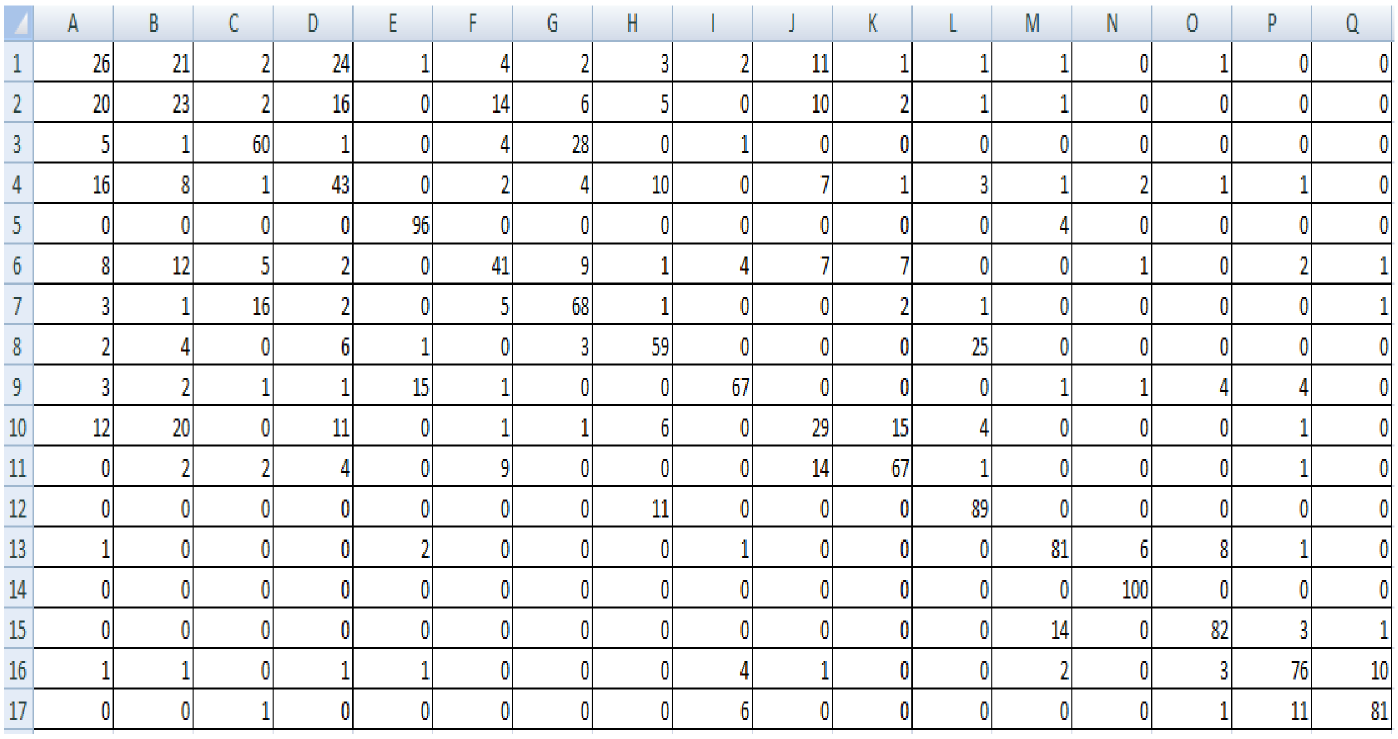
Confusion Matrix

Train: Table 3.2.8



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	98	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	99	0	0	0	1	0	0	0	0	0	0	0	0	0	0
4	1	1	0	97	0	0	0	1	0	0	0	0	0	0	0	0	0
5	0	0	0	0	99	0	0	0	0	0	0	0	0	1	0	0	0
6	0	1	0	0	0	99	0	0	0	0	0	0	0	0	0	0	0
7	0	0	1	0	0	0	99	0	0	0	0	0	0	0	0	0	0
8	0	1	1	2	0	0	0	94	0	0	0	2	0	0	0	0	0
9	0	0	0	0	1	0	0	0	99	0	0	0	0	0	0	0	0
10	3	5	0	1	0	0	0	1	0	90	0	0	0	0	0	0	0
11	0	1	0	0	0	2	0	0	0	0	96	1	0	0	0	0	0
12	0	0	0	1	0	0	0	2	0	0	0	97	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	99	0	1	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0
15	0	1	1	0	0	0	0	0	0	0	0	0	4	0	94	0	0
16	0	0	0	0	0	0	1	0	1	0	0	0	0	0	2	93	3
17	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	1	96

Test: Table 3.2.9

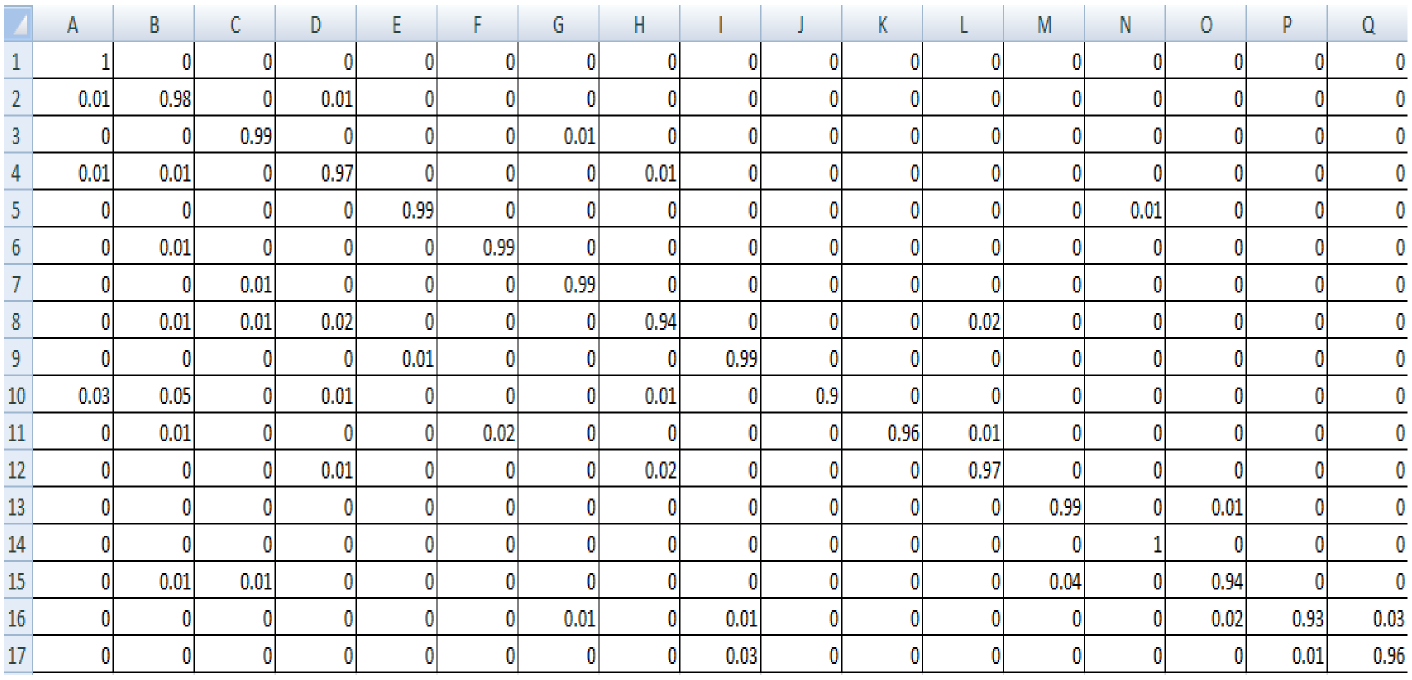


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	26	21	2	24	1	4	2	3	2	11	1	1	1	0	1	0	0
2	20	23	2	16	0	14	6	5	0	10	2	1	1	0	0	0	0
3	5	1	60	1	0	4	28	0	1	0	0	0	0	0	0	0	0
4	16	8	1	43	0	2	4	10	0	7	1	3	1	2	1	1	0
5	0	0	0	0	96	0	0	0	0	0	0	0	4	0	0	0	0
6	8	12	5	2	0	41	9	1	4	7	7	0	0	1	0	2	1
7	3	1	16	2	0	5	68	1	0	0	2	1	0	0	0	0	1
8	2	4	0	6	1	0	3	59	0	0	0	25	0	0	0	0	0
9	3	2	1	1	15	1	0	0	67	0	0	0	1	1	4	4	0
10	12	20	0	11	0	1	1	6	0	29	15	4	0	0	0	1	0
11	0	2	2	4	0	9	0	0	0	14	67	1	0	0	0	1	0
12	0	0	0	0	0	0	0	11	0	0	0	89	0	0	0	0	0
13	1	0	0	0	2	0	0	0	1	0	0	0	81	6	8	1	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	14	0	82	3	1
16	1	1	0	1	1	0	0	0	4	1	0	0	2	0	3	76	10
17	0	0	1	0	0	0	0	0	6	0	0	0	0	0	1	11	81

Wallpaper

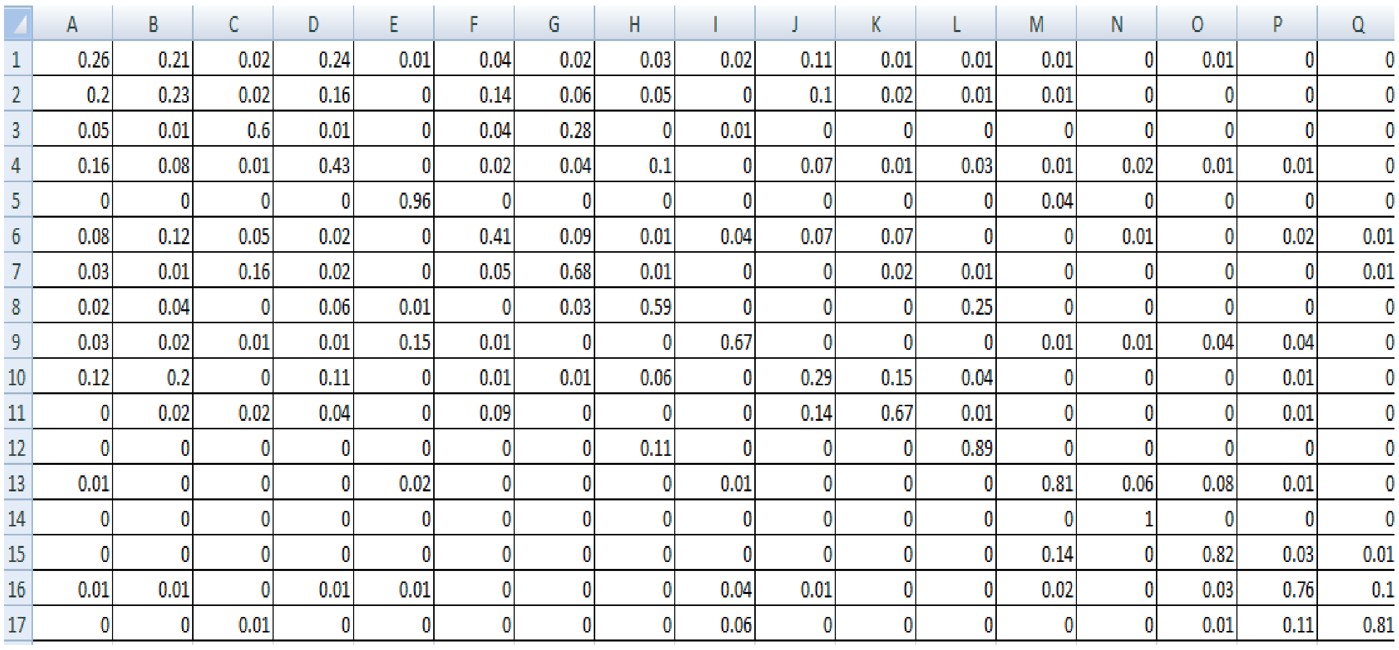
Classification Matrix:

Train Table 3.2.10



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0.01	0.98	0	0.01	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0.99	0	0	0	0.01	0	0	0	0	0	0	0	0	0	0
4	0.01	0.01	0	0.97	0	0	0	0.01	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0.99	0	0	0	0	0	0	0	0	0.01	0	0	0
6	0	0.01	0	0	0	0.99	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0.01	0	0	0	0.99	0	0	0	0	0	0	0	0	0	0
8	0	0.01	0.01	0.02	0	0	0	0.94	0	0	0	0.02	0	0	0	0	0
9	0	0	0	0	0.01	0	0	0	0.99	0	0	0	0	0	0	0	0
10	0.03	0.05	0	0.01	0	0	0	0.01	0	0.9	0	0	0	0	0	0	0
11	0	0.01	0	0	0	0.02	0	0	0	0	0.96	0.01	0	0	0	0	0
12	0	0	0	0.01	0	0	0	0.02	0	0	0	0.97	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0.99	0	0.01	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
15	0	0.01	0.01	0	0	0	0	0	0	0	0	0	0.04	0	0.94	0	0
16	0	0	0	0	0	0	0.01	0	0.01	0	0	0	0	0	0.02	0.93	0.03
17	0	0	0	0	0	0	0	0	0.03	0	0	0	0	0	0	0.01	0.96

Test: Table 3.2.11

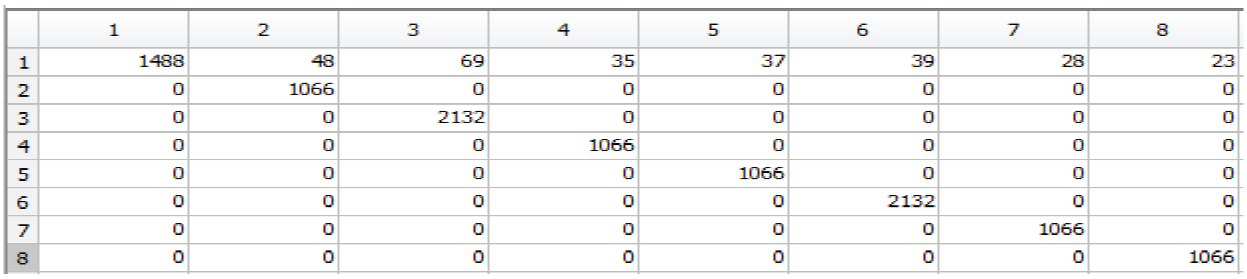


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	0.26	0.21	0.02	0.24	0.01	0.04	0.02	0.03	0.02	0.11	0.01	0.01	0.01	0	0.01	0	0
2	0.2	0.23	0.02	0.16	0	0.14	0.06	0.05	0	0.1	0.02	0.01	0.01	0	0	0	0
3	0.05	0.01	0.6	0.01	0	0.04	0.28	0	0.01	0	0	0	0	0	0	0	0
4	0.16	0.08	0.01	0.43	0	0.02	0.04	0.1	0	0.07	0.01	0.03	0.01	0.02	0.01	0.01	0
5	0	0	0	0	0.96	0	0	0	0	0	0	0	0.04	0	0	0	0
6	0.08	0.12	0.05	0.02	0	0.41	0.09	0.01	0.04	0.07	0.07	0	0	0.01	0	0.02	0.01
7	0.03	0.01	0.16	0.02	0	0.05	0.68	0.01	0	0	0.02	0.01	0	0	0	0	0.01
8	0.02	0.04	0	0.06	0.01	0	0.03	0.59	0	0	0	0.25	0	0	0	0	0
9	0.03	0.02	0.01	0.01	0.15	0.01	0	0	0.67	0	0	0	0.01	0.01	0.04	0.04	0
10	0.12	0.2	0	0.11	0	0.01	0.01	0.06	0	0.29	0.15	0.04	0	0	0	0.01	0
11	0	0.02	0.02	0.04	0	0.09	0	0	0	0.14	0.67	0.01	0	0	0	0.01	0
12	0	0	0	0	0	0	0	0.11	0	0	0	0.89	0	0	0	0	0
13	0.01	0	0	0	0.02	0	0	0	0.01	0	0	0	0.81	0.06	0.08	0.01	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0.14	0	0.82	0.03	0.01
16	0.01	0.01	0	0.01	0.01	0	0	0	0.04	0.01	0	0	0.02	0	0.03	0.76	0.1
17	0	0	0.01	0	0	0	0	0	0.06	0	0	0	0	0	0.01	0.11	0.81

Taiji

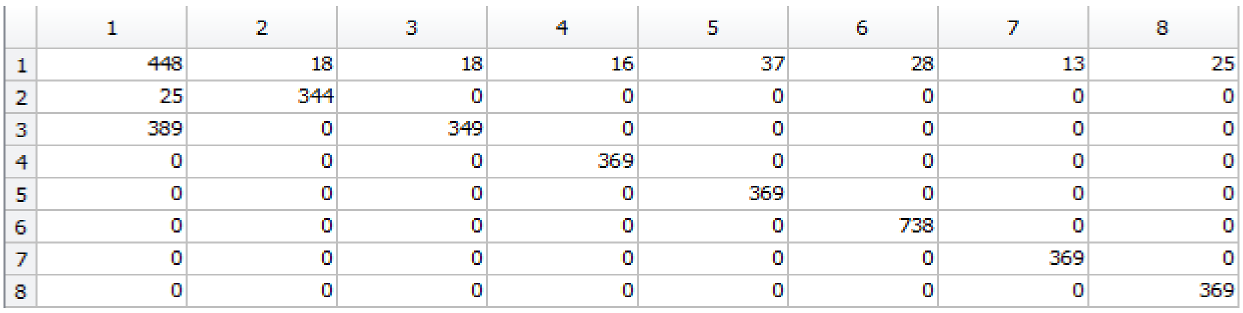
Confusion Matrix

Train: Table 3.2.12



	1	2	3	4	5	6	7	8
1	1488	48	69	35	37	39	28	23
2	0	1066	0	0	0	0	0	0
3	0	0	2132	0	0	0	0	0
4	0	0	0	1066	0	0	0	0
5	0	0	0	0	1066	0	0	0
6	0	0	0	0	0	2132	0	0
7	0	0	0	0	0	0	1066	0
8	0	0	0	0	0	0	0	1066

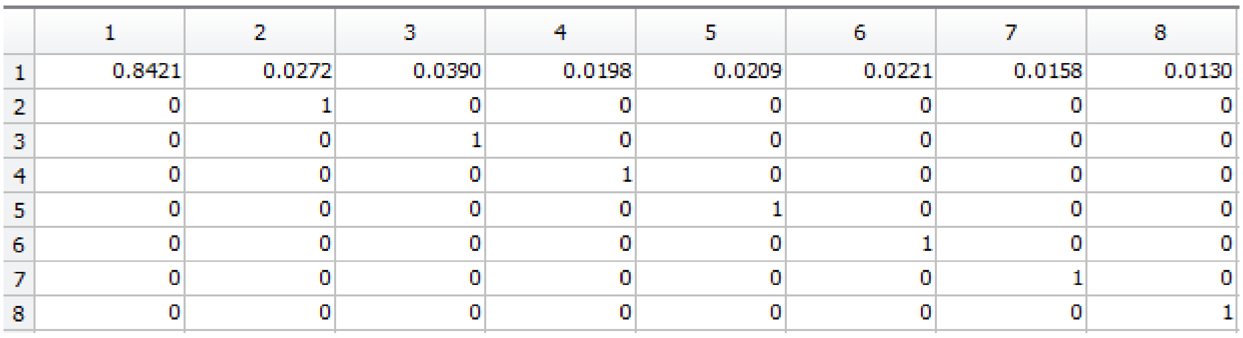
Test : Table 3.2.13



	1	2	3	4	5	6	7	8
1	448	18	18	16	37	28	13	25
2	25	344	0	0	0	0	0	0
3	389	0	349	0	0	0	0	0
4	0	0	0	369	0	0	0	0
5	0	0	0	0	369	0	0	0
6	0	0	0	0	0	738	0	0
7	0	0	0	0	0	0	369	0
8	0	0	0	0	0	0	0	369

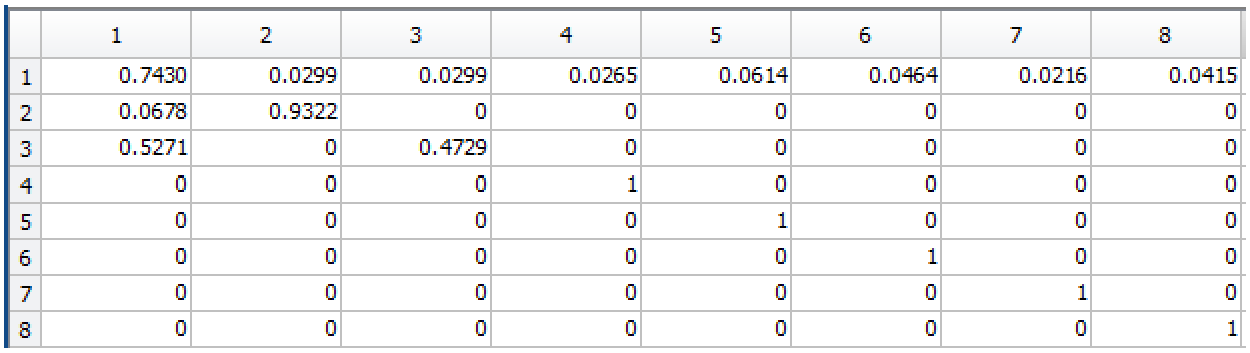
Classification Matrix

Train : Table 3.2.14



	1	2	3	4	5	6	7	8
1	0.8421	0.0272	0.0390	0.0198	0.0209	0.0221	0.0158	0.0130
2	0	1	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0
4	0	0	0	1	0	0	0	0
5	0	0	0	0	1	0	0	0
6	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	1

Test: Table 3.2.15



	1	2	3	4	5	6	7	8
1	0.7430	0.0299	0.0299	0.0265	0.0614	0.0464	0.0216	0.0415
2	0.0678	0.9322	0	0	0	0	0	0
3	0.5271	0	0.4729	0	0	0	0	0
4	0	0	0	1	0	0	0	0
5	0	0	0	0	1	0	0	0
6	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	1

If we compare least square test classification matrix for taiji(Table 3.1.13) and that of fisher +KNN (Table 3.2.15), we will see first row filled with coefficients. It means that both classifiers misclassifying other class observations to Class '0'. This can be considered as limitation of Linear classifiers.

4. Conclusion:

1. Least square classifier gave comparatively good results on all three datasets. With increase in number of observations and total number of classes, least square accuracy decreased. Least square has problem of outlier. If you add extra point away from decision boundary, it rotates decision boundary towards that point which can cause multiple misclassification. This issue also arises binary target vector has distribution which is far away from Gaussian.

2. Fisher discriminant analysis is one the important dimensionality reduction technique because it not only project data from high dimension to lower dimension, but also take measure against overlapping of classes. Because Fisher analysis provides weight parameters depend on mean difference between classes and within class variance. Using fisher analysis, dimensionality can be reduce till one less than total number of classes. If you try to further reduce dimensionality, information can be lost while projection of data from high dimension to lower dimension.

3. K nearest neighbor classifier's non parametric approach somewhat overshadowed significance of Fisher's discriminator analysis. Still KNN gave better results as compare to least square on all datasets. More observations gives liberty to tune higher K value with better testing accuracy.

4. Linear classifier performed well on less observations, less features or more observations less classes. But it significantly failed to produce good testing results on more features and comparatively low observations. So probably these types of datasets need to be classify by using non linear classifier.

5. References:

[1] https://www-users.cs.umn.edu/~kumar001/dmbook/ch4.pdf

[2] http://web.mit.edu/6.S097/www/resources/L01.pdf

[3] Pattern Recognition by Bishop

[4] Fisher Linear Discriminant Analysis, Cheng Li, Bingyu Wang, August 31, 2014

[5] Wikipedia

[6] Stanford CS231n Convolution neural net lecture notes.

[7] Least Squares Classification by Stephen Boyd EE103, Stanford University
November 4, 2017