LAB MANUAL

ACADEMIC YEAR: 2012-2013

DEPARTMENT: COMPUTER ENGINEERING

CLASS: T.E. **SEMESTER:** II

SUBJECT: Pattern Recognition(Elective-3)

INDEX OF LAB ASSIGNMENTS

Revised on: 13/12/12

SUBJECT: Pattern Recognition

Student should formulate problem definition for given learning objectives and check it from the lab teacher. Modify and recheck if required. Implement and submit the same as lab assignments.

Assign. No.	**Problem Definition/Statement/Topic**	**Completion date**
1	**Design of Bayesian Classifier**	3rd week of Dec
2	**Study and recognition of different patterns**	2nd week of Jan
3	**Implementation of Edit distance problem**	1st week of Feb
4	**Study of different color spaces and color based image segmentation**	4th week of Feb
5	**Analysis of different feature descriptor**	2nd week of March
6	**Experimentation of PCA and ICA, Interpretation of ROC**	4th week of March

Subject Coordinator **Head of Department**

(Prof. A.S.Ghotkar) **(Computer Department)**

TITLE	**Bayesian Classifier**
PROBLEM STATEMENT/ DEFINITION	**To design of Bayesian classifier for any Image processing application /statistical decision theory**
OBJECTIVE	1.To understand concept of Bayesian classifier 2.To build a Bayesian classifier for identified application
S/W PACKAGES AND HARDWARE APPARATUS USED	VC++/JAVA with OpenCV library 2.0 and above, Windows XP/7, PC with the configuration as Pentium IV and above, 1 GB RAM, 120 G.B HDD, 15''Color Monitor, Keyboard, Mouse
REFERENCES	R.O.Duda, "Pattern Classification" K.Fukunaga, "Statistical Pattern Recognition" O'reilley, " Learning OpenCV"
STEPS	• Identification of application • Collection of data • Design of algorithm • Identification of openCV library function • Coding • Testing and analysis
INSTRUCTIONS FOR WRITING JOURNAL	Problem Definition Scope and limitation Theory OpenCv Library function used Algorithm Testing and Analysis Program Listing Output Conclusion References

Aim : To Understand the concept of Bayesian classifier and Implement for any Image processing application /statistical decision problem.

Prerequisites:

- Probability ,Statistical theory
- Concepts of Pattern recognition

Learning Objectives:

To understand and implement using OpenCV in VC++/Java

- Identification of class, features for identified problem
- Design of Bayesian for identified problem
- Likelihood ratio tests
- Calculation of probability error, loss function and Risk

Learning Outcomes :

The student will be able to

- Design and Implement Bayesian Classification for Identified problem
- Calculation of Probability error, loss function.

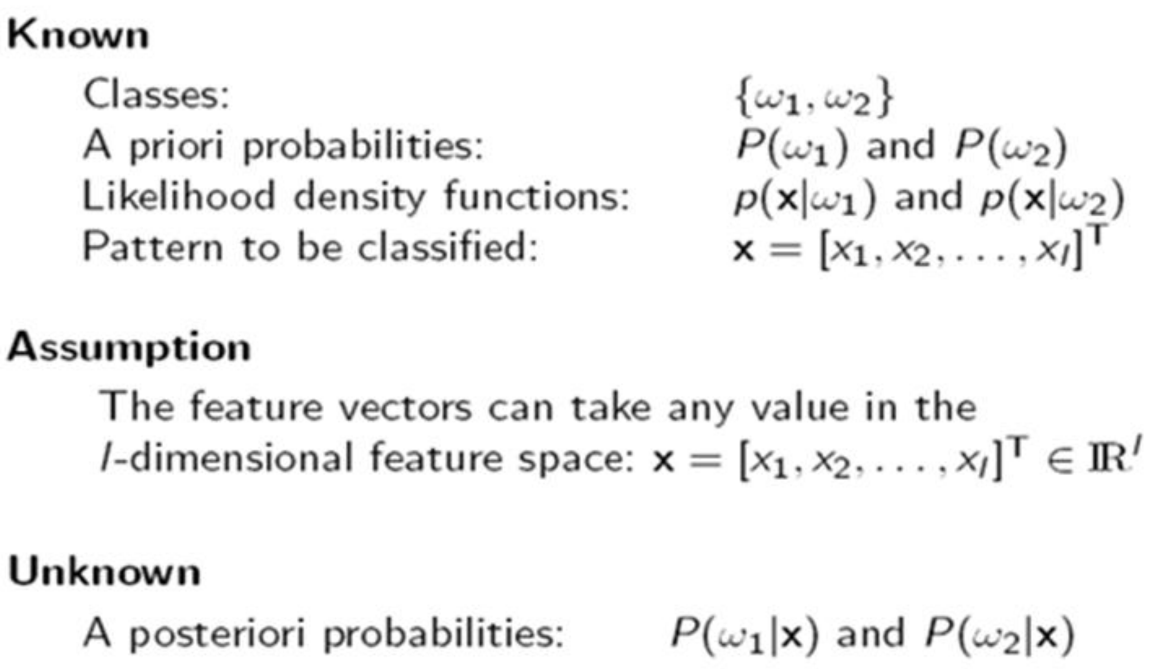
Theory/Concept:

Frequentist or Bayesian approach to pattern recognition?

The first pattern classifier – the linear discriminant presented by Fisher – was developed in the Frequentist tradition. The frequentist approach entails that the model parameters are considered unknown, but objective. The parameters are then computed (estimated) from the collected data. For the linear discriminant, these parameters are precisely the mean vectors and the Covariance matrix. Also the probability of each class $p(\mathrm{label}|\boldsymbol{\theta})$is estimated from the collected dataset. Note that the usage of 'Bayes rule' in a pattern classifier does not make the classification approach Bayesian.

Bayesian statistics has its origin in Greek philosophy where a distinction was already made between the 'a priori' and the 'a posteriori' knowledge. Later Kant defined his distinction between what is a priori known – before observation – and the empirical knowledge gained from observations. In a Bayesian pattern classifier, the class probabilities $p(\mathrm{label}|\boldsymbol{\theta})$can be chosen by the user, which are then a priori. Moreover, experience quantified as a priori parameter values can be weighted with empirical observations – using e.g. the Beta-(conjugate prior) and Dirichlet-distributions. The Bayesian approach facilitates a seamless intermixing between expert knowledge in the form of subjective probabilities, and objective observations.

Probabilistic pattern classifiers can be used according to a frequentist or a Bayesian approach.



Known

Classes:	$\{\omega_1, \omega_2\}$
A priori probabilities:	$P(\omega_1)$ and $P(\omega_2)$
Likelihood density functions:	$p(\mathbf{x}\|\omega_1)$ and $p(\mathbf{x}\|\omega_2)$
Pattern to be classified:	$\mathbf{x} = [x_1, x_2, \ldots, x_l]^T$

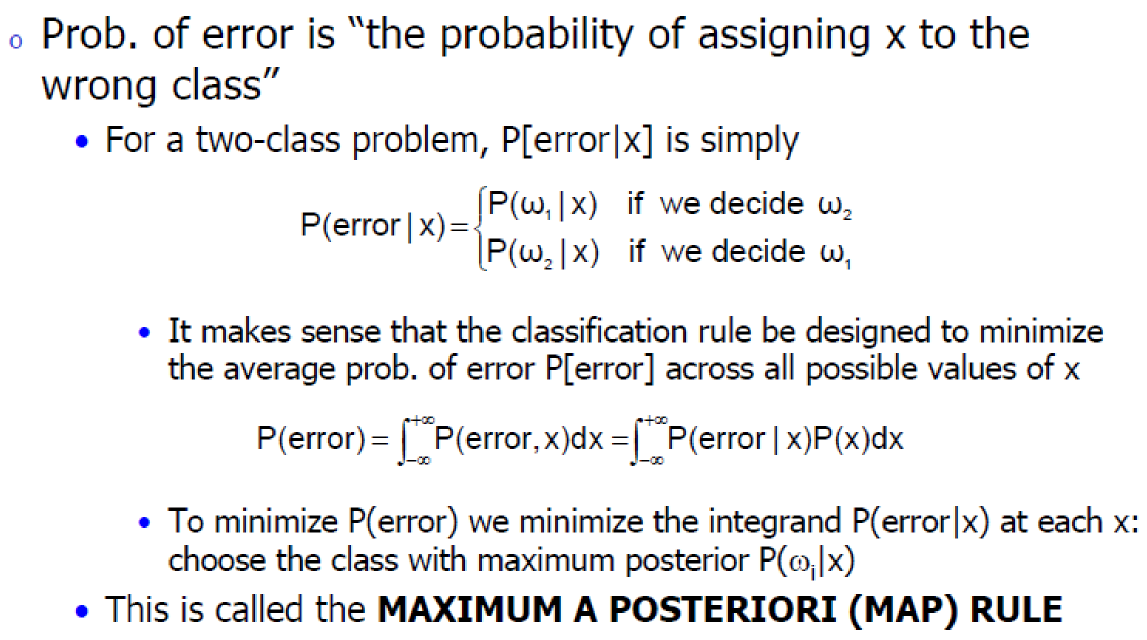
Assumption

The feature vectors can take any value in the l-dimensional feature space: $\mathbf{x} = [x_1, x_2, \ldots, x_l]^T \in \mathbb{R}^l$

Unknown

A posteriori probabilities: $P(\omega_1|\mathbf{x})$ and $P(\omega_2|\mathbf{x})$

Probability Error:



- Prob. of error is "the probability of assigning x to the wrong class"
  - For a two-class problem, P[error|x] is simply

$$P(\text{error}\,|\,x) = \begin{cases} P(\omega_1\,|\,x) & \text{if we decide } \omega_2 \\ P(\omega_2\,|\,x) & \text{if we decide } \omega_1 \end{cases}$$

    - It makes sense that the classification rule be designed to minimize the average prob. of error P[error] across all possible values of x

$$P(\text{error}) = \int_{-\infty}^{+\infty} P(\text{error}, x)dx = \int_{-\infty}^{+\infty} P(\text{error}\,|\,x)P(x)dx$$

    - To minimize P(error) we minimize the integrand P(error|x) at each x: choose the class with maximum posterior $P(\omega_i|x)$
  - This is called the **MAXIMUM A POSTERIORI (MAP) RULE**

Variation on LRT:

- The LRT that minimizes the Bayes Risk is called the Bayes Criterion

$$\Lambda(x)=\frac{P(x|\omega_1)}{P(x|\omega_2)} \underset{\omega_2}{\overset{\omega_1}{\gtrless}} \frac{(C_{12}-C_{22})P[\omega_2]}{(C_{21}-C_{11})P[\omega_1]}$$ **Bayes criterion**

- Many times we will simply be interested in minimizing P[error], which is a special case of the Bayes Criterion if we use a zero-one cost function
  - This version of the LRT is referred to as the Maximum A Posteriori Criterion, since it seeks to maximize the posterior $P(\omega_i|x)$

$$C_{ij}=\begin{cases}0 & i=j\\ 1 & i\neq j\end{cases} \Rightarrow \Lambda(x)=\frac{P(x|\omega_1)}{P(x|\omega_2)} \underset{\omega_2}{\overset{\omega_1}{\gtrless}} \frac{P(\omega_2)}{P(\omega_1)} \Leftrightarrow \frac{P(\omega_1|x)}{P(\omega_2|x)} \underset{\omega_2}{\overset{\omega_1}{\gtrless}} 1$$ **Maximum A Posteriori (MAP) Criterion**

- Finally, for the case of equal priors $P[\omega_i]=1/2$ and zero-one cost function, the LTR is called the Maximum Likelihood Criterion, since it will maximize the likelihood $P(x|\omega_i)$

$$\left.\begin{matrix} C_{ij}=\begin{cases}0 & i=j\\ 1 & i\neq j\end{cases} \\ P(\omega_i)=\frac{1}{C} \quad \forall i\end{matrix}\right\} \Rightarrow \Lambda(x)=\frac{P(x|\omega_1)}{P(x|\omega_2)} \underset{\omega_2}{\overset{\omega_1}{\gtrless}} 1$$ **Maximum Likelihood (ML) Criterion**

Case Study : Apply Bayesian classification for skin detection.

Steps for skin detection using Bayesian Classification:[1]

- A more appropriate measure for skin detection would be P(skin|c)- a probability of observing skin, given a concrete c color value.
- To compute this probability, the Bayes rule is used:

$$P(skin|c)=\frac{P(c|skin)P(skin)}{P(c|skin)P(skin)+P(c|\neg skin)P(\neg skin)}$$

- P(c|skin) and P(c|¬skin) are directly computed from skin and non-skin color histogram
- The prior probabilities P(skin) and P(¬skin) can also be estimated from the overall number of skin and non-skin samples in the training set.
- For skin detection decision rule choose the value of Θ(threshold value)such that

  P(skin|c)>= Θ

[1].Valdimir Vezhnevets,Vassili Sazonov,AllaAndreeva, “A survey on Pixel-based Skin Color Detection Techniques”

Description using set theory

Let S={ I,O,F,S,E}
Where I=set of Input={ skin and non-skin samples }
Output= set of Output={Skin color samples }

F= set of functions={ create histogram, calculate P(skin|c), Decision rule (P(skin|c)>= Θ) }
S=set of success= { correct skin classification }
E=set of exception={ misclassification }

Testing and Analysis:

1. Test the LRT for given sample data.

FAQs

1) Is Bayesian classification used in both parametric and non-parametric approaches.
   Ans: yes. It can. Only the calculation of likelihood density function changed in both approaches.
2) How Bayesian rule will be applicable for multiple features?
   Ans: Navies bays solution is there.

Review Questions

1) What is mean by pattern, feature, classes?
2) What is pattern classification?
3) What is class conditional, priori and posteriori probability?
4) What is mean by discriminant function?
5) What is Bayes risk?
6) What is Naïve bays classification.

Practice assignments

1) Design and Implement Bayesian classification for object classification in given image.
2) Design and implement Naïve bays classification for any statistical data(refer class notes: case study :-play tennis or not).

Revised on:13/12/12

TITLE	**Study and recognition of different patterns**
PROBLEM STATEMENT/ DEFINITION	Write a Program for recognition of line patterns.
OBJECTIVE	To Study line patterns in a given set of points. To write program for to recognize line patterns in a given set of points
S/W PACKAGES AND HARDWARE APPARATUS USED	VC++/JAVA with OpenCV library, PC with minimum e configuration as Pentium IV 1.7 GHz. 128M.B RAM, 40 G.B HDD, 15"Color Monitor, Keyboard, Mouse
REFERENCES	R.O.Duda, “Pattern Classification” K.Fukunaga, “Statistical Pattern Recognition” O’reilley, “ Learning OpenCV”
STEPS	Identification of application Collection of data Design of algorithm Identification of openCV library function Coding Testing and analysis
INSTRUCTIONS FOR WRITING JOURNAL	Problem Definition Scope and limitation Theory OpenCv Library function used Algorithm Testing and Analysis Program Listing Output Conclusion

Aim : Write a Program for recognition of line patterns.

Prerequisites:

- Concept of features in object recognition
- Different types of features

Learning Objectives:

- To Study line patterns in a given set of points.
- To write program for to recognize line patterns in a given set of points

Learning Outcomes :

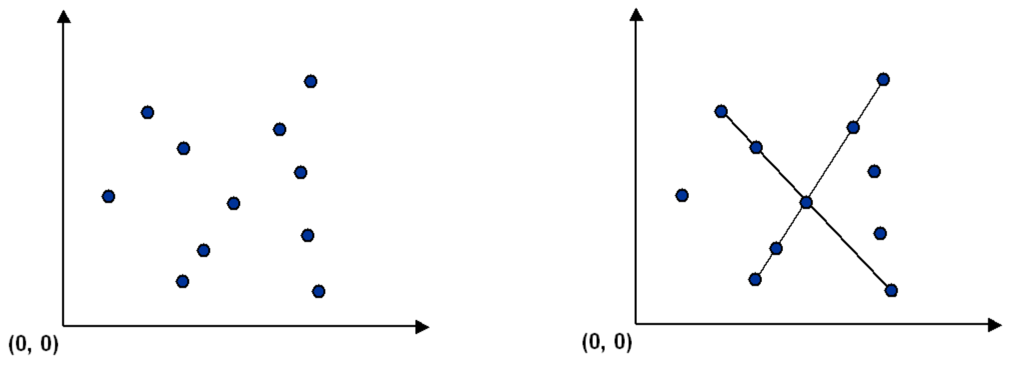
The student will be able to

- Use OpenCv Library functions for recognition of line from given set of point.
- Implement recognition of line patterns in give image dataset

Theory/Concept:

Computer vision involves analyzing patterns in visual images and reconstructing the real world objects that produced them. The process in often broken up into two phases: *feature detection* and *pattern recognition*. Feature detection involves selecting important features of the image; pattern recognition involves discovering patterns in the features. We will investigate a particularly clean pattern recognition problem involving points and line segments. This kind of pattern recognition arises in many other applications, for example statistical data analysis.

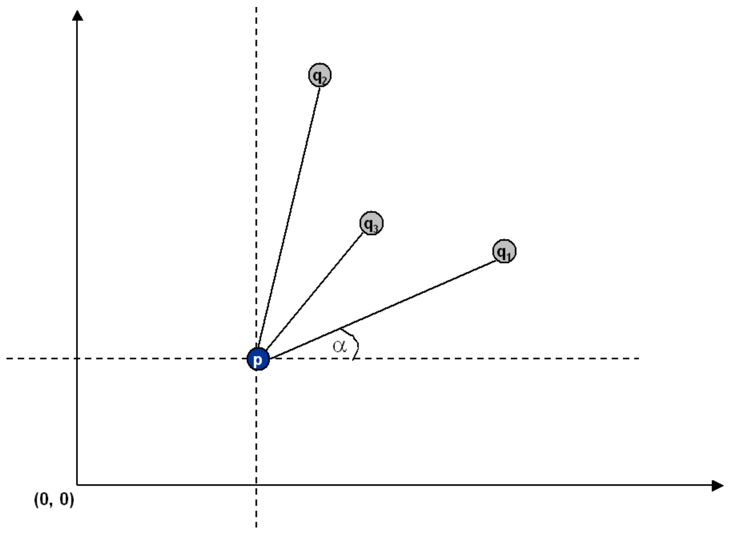
The problem. Given a set of N feature points in the plane, determine every line segment that contains 4 or more of the points, and plot all such line segments.



Brute force. Write a program `Brute.java` that examines 4 points at a time and checks if they all lie on the same line segment, plotting any such line segments in turtle graphics. To get started, you may use the data type Point.java and the client program PointPlotter.java which reads in a list of points from standard input and plots them using Turtle graphics. You will need to supply additional interface functions to `Point.java` in order to support the brute force client, e.g., checking whether three points lie on the same line.

A sorting solution. Remarkably, it is possible to solve the problem much faster than the brute force solution described above. Given a point p, the following method determines whether p participates in a group of 4 or more collinear points. Applying this method for each of the N points in turn yields an efficient algorithm to the problem.

- Think of p as the origin.
- For each point q, determine the angle it makes with p.
- Sort the points according to the angle each makes with p.
- Check if any 3 adjacent points in the sorted order have equal angles with p. If so,these points, together with p, are collinear.



Write a program `Fast.java` that implements this algorithm. The algorithm solves the problem because points that make the same angle with p are collinear, and sorting brings such points together. The algorithm is fast because the bottleneck operation is sorting.

Point data type. Create an immutable data type `Point` that represents a point in the plane by implementing the following API:

```
public class Point implements Comparable<Point> {
   public final Comparator<Point> SLOPE_ORDER;        // compare points by
slope to this point

   public Point(int x, int y)                         // construct the
point (x, y)

   public   void draw()                               // draw this point
   public   void drawTo(Point that)                   // draw the line
segment from this point to that point
   public String toString()                           // string
representation

   public    int compareTo(Point that)                // is this point
lexicographically smaller than that point?
   public double slopeTo(Point that)                  // the slope between
this point and that point
}
```

To get started, use the data type Point.java, which implements the constructor and the `draw()`, `drawTo()`, and `toString()` methods. Your job is to add the following components.

- The `compareTo()` method should compare points by their *y*-coordinates, breaking ties by their *x*-coordinates. Formally, the invoking point (x_0, y_0) is *less than* the argument point (x_1, y_1) if and only if either $y_0 < y_1$ or if $y_0 = y_1$ and $x_0 < x_1$.
- The `slopeTo()` method should return the slope between the invoking point (x_0, y_0) and the argument point (x_1, y_1), which is given by the formula $(y_1 - y_0) / (x_1 - x_0)$. Treat the slope of a horizontal line segment as positive zero; treat the slope of a vertical line segment as positive infinity; treat the slope of a degenerate line segment (between a point and itself) as negative infinity.
- The `SLOPE_ORDER` comparator should compare points by the slopes they make with the invoking point (x_0, y_0). Formally, the point (x_1, y_1) is *less than* the point (x_2, y_2) if and only if the slope $(y_1 - y_0) / (x_1 - x_0)$ is less than the slope $(y_2 - y_0) / (x_2 - x_0)$. Treat horizontal, vertical, and degenerate line segments as in the `slopeTo()` method.

Input format. The data file consists of an integer N, followed by N pairs of integers (x, y) between 0 and 32,768.

```
5
16384  19200
16384  21120
16384  32000
16384  21761
10000  10000
```

Output format. Your program should use Turtle graphics to draw all of the points and the line segments your program discovers. Using the point data type supplied, the command `p.draw()` draws the point p and the command `p.drawTo(q)` draws the line segment from p to q. Note that the points are scaled down by a factor of 64.0 so that they fit snugly inside a 512-by-512 turtle graphics window.

Analysis. Estimate the running time of your two programs as a function of N. Provide analytical and empirical evidence to support your answer.

FAQs

1. In what application line features plays important role in image processing application?
   Ans: For any image processing application where line is the part of the object generally contour based object recognition. E.g. OCR, Gesture recognition, medical image processing, satellite image processing, image enhancement application.

Review Questions

1. How to define line with set of points.
2. What is the role of line detection in object recognition?
3. What are the different OpenCV library function will help to detect the line?

Practice assignments

1. Write a program for identification of corner as a feature in a given image.
2. Write a program for identification of circle as a feature in a given image.

TITLE	Edit Distance Problem
PROBLEM STATEMENT/ DEFINITION	Implementation of Edit distance problem
OBJECTIVE	To understand Edit Distance Problem To implement Edit Distance Problem for different applications
S/W PACKAGES AND HARDWARE APPARATUS USED	VC++/JAVA with OpenCV library, PC with minimum e configuration as Pentium IV 1.7 GHz. 128M.B RAM, 40 G.B HDD, 15"Color Monitor, Keyboard, Mouse
REFERENCES	R.O.Duda, “Pattern Classification” K.Fukunaga, “Statistical Pattern Recognition” O’reilley, “ Learning OpenCV”
STEPS	Identification of application Collection of data Design of algorithm Identification of openCV library function Coding Testing and analysis
INSTRUCTIONS FOR WRITING JOURNAL	Problem Definition Scope and limitation Theory OpenCv Library function used Algorithm Testing and Analysis Program Listing Output Conclusion

Aim: Implement spell checker for given dictionary file containing all the correct words. Then take a word as input from the user and find the closest 5 matches of that word in the dictionary.

Prerequisites:

- Concept of distance problem.
- Use of string matching.

Learning Objectives:

- To Study Edit distance problem.
- To apply edit distance problem for string matching problem considering string as a pattern.

Learning Outcomes :

The student will be able to

- Implement Edit distance problem for string matching.
- Display no. of. Mismatch character.

Theory/Concept:

In the edit distance problem, a pattern consists of a set of ordered symbols. Each symbol is a feature vector. If these symbols are letters then the pattern is a word. Such problems arise in automatic editing and text retrieval applications. A string is read and matched against a set of known strings (or, dictionary). Then the task is to find the best match. For example the edit distance problem arises in the spell checker of a word processor.

There are three basic types of errors:

- Wrong symbol ("be***f***uty" instead of "beauty")
- Insertion error ("bea***r***uty")
- Deletion error ("beuty")

The **edit distance** between two string patterns A and B, denoted by D (A, B), is defined as the minimum total number of changes C, insertions I and deletions R required to transform A into B.

$D(A, B) = \min_j [C(j) + I(j) + R(j)]$
Where j runs over all possible combinations of symbol variations in order to obtain B from A.

Optimal path searching by Dynamic Programming:
Define a cost function D (i, j) to be the cost up to the i-th reference symbol and j-th test symbol. The constrains are:

- D (0, 0) = 0
- A complete path is searched
- Each node (i, j) can be reached through only three predecessors: [See **class notes**]
    - (i-1, j)
    - (i-1, j-1)
    - (i, j-1)

- Costs associated with the above three transitions are:
    - Diagonal Transition:
        - $d(i, j| i-1, j-1) = 0$ if $ref(i) = test(j)$
          $= 1$ if $ref(i) <> test(j)$
    - Horizontal or Vertical Transition:
        - $d(i, j| i-1, j) = d(i, j| i, j-1) = 1$

Algorithm Edit-Distance (ref(1..I), test(1..J)):

```
1. D(0, 0) = 0
2. for i=1 to I do
      a. D(i, 0) = D(i-1, 0) + 1
3. for j=1 to J do
      a. D(0, j) = D(0, j-1) + 1
4. for i=1 to I do
      a. for j=1 to J do
            I.   c1 = D(i-1, j-1) + d(i, j| i-1, j-1)
            II.  c2 = D(i-1, j) + 1
            III. c3 = D(i, j-1) + 1
            IV.  D(i,j) = min (c1, c2, c3)
5. return D(I, J)
```

FAQs

1. What are the different application where we can use Edit distance problem
   Ans: In different template matching application such as spell checker in text editor, Optical character recognition, speech processing etc.
2. Does Edit distance problem will help to match image patterns?
   Ans: yes. If features are described in to string/number template.

Review Questions

1. How to use Edit distance problem in image pattern?
2. What are the different algorithms for string matching.?

Revised on:13/12/12

TITLE	Color Spaces and color based image segmentation
PROBLEM STATEMENT/ DEFINITION	Study of different color spaces and color based image segmentation
OBJECTIVE	To study different color spaces. To Implement and analysis color-based image segmentation using different color spaces.
S/W PACKAGES AND HARDWARE APPARATUS USED	VC++/JAVA with OpenCV library, PC with minimum e configuration as Pentium IV 1.7 GHz. 128M.B RAM, 40 G.B HDD, 15"Color Monitor, Keyboard, Mouse
REFERENCES	R.O.Duda, “Pattern Classification” K.Fukunaga, “Statistical Pattern Recognition” O’reilley, “ Learning OpenCV”
STEPS	Identification of application Collection of data Design of algorithm Identification of openCV library function Coding Testing and analysis
INSTRUCTIONS FOR WRITING JOURNAL	Problem Definition Scope and limitation Theory OpenCv Library function used Algorithm Testing and Analysis Program Listing Output Conclusion

Aim: Study of different color spaces and color based Image segmentation

Prerequisites:

- Concept of Image segmentation.
- Different color models

Learning Objectives:

- To Study Different color spaces for image segemnatation.
- Implement and Analysis Image segmentation using different color spaces.

Learning Outcomes :

The student will be able to

- Compare different color spaces and their use.
- Implement color based image segmentation.

Theory/Concept:

Color image segmentation is useful in many applications.From the segmentation results, it is possible to identify regions of interest and objects in the scene, which is very beneficial to the subsequent image analysis or annotation.

A color model is an abstract mathematical model describing the way colors can be represented as tuples of numbers, typically as three or four values or *color components* (e.g. RGB and CMYK are color models). However, a color model with no associated mapping function to an absolute color space is a more or less arbitrary color system with no connection to any globally understood system of color interpretation.

Adding a certain mapping function between the color model and a certain reference color space results in a definite "footprint" within the reference color space. This "footprint" is known as a gamut, and, in combination with the color model, defines a new **color space**. For example, Adobe RGB and sRGB are two different absolute color spaces, both based on the RGB model.

RGB uses additive color mixing, because it describes what kind of *light* needs to be *emitted* to produce a given color. Light is added together to create form from out of the darkness. RGB stores individual values for red, green and blue. RGBA is RGB with an additional channel, alpha, to indicate transparency.

Common color spaces based on the RGB model include sRGB, Adobe RGB and ProPhoto RGB.

CMYK uses subtractive color mixing used in the printing process, because it describes what kind of inks need to be applied so the light *reflected* from the substrate and through the inks produces a given color. One starts with a white substrate (canvas, page, etc.), and uses ink to subtract color from white to create an image. CMYK stores ink values for cyan, magenta, yellow and black. There are many CMYK color spaces for different sets of inks, substrates, and press characteristics (which change the dot gain or transfer function for each ink and thus change the appearance).

YIQ was formerly used in NTSC (North America, Japan and elsewhere) television broadcasts for historical reasons. This system stores a luminance value with two chrominance values, corresponding approximately to the amounts of blue and red in the color. It is similar to the YUV scheme used in most video capture systems[2] and in PAL (Australia, Europe, except France, which uses SECAM) television, except that the YIQ color space is rotated 33° with respect to the YUV color space. The YDbDr scheme used by SECAM television is rotated in another way.

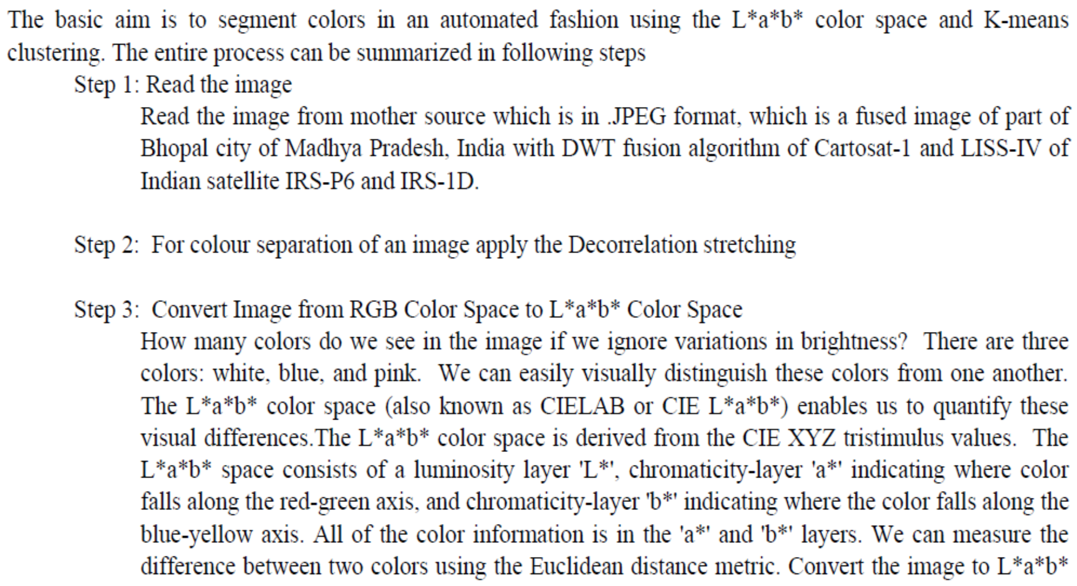
YPbPr is a scaled version of YUV. It is most commonly seen in its digital form, YCbCr, used widely in video and image compression schemes such as MPEG and JPEG.

xvYCC is a new international digital video color space standard published by the IEC (IEC 61966-2-4). It is based on the ITU BT.601 and BT.709 standards but extends the gamut beyond the R/G/B primaries specified in those standards.

HSV (**h**ue, **s**aturation, **v**alue), also known as HSB (hue, saturation, **b**rightness) is often used by artists because it is often more natural to think about a color in terms of hue and saturation than in terms of additive or subtractive color components. HSV is a transformation of an RGB colorspace, and its components and colorimetry are relative to the RGB colorspace from which it was derived.

HSL (**h**ue, **s**aturation, **l**ightness/**l**uminance), also known as HLS or HSI (hue, saturation, **i**ntensity) is quite similar to HSV, with "lightness" replacing "brightness". The difference is that the *brightness* of a pure color is equal to the brightness of white, while the *lightness* of a pure color is equal to the lightness of a medium gray.

Color based segmentation using K-mean clustering Algorithm [1]


The basic aim is to segment colors in an automated fashion using the L*a*b* color space and K-means clustering. The entire process can be summarized in following steps

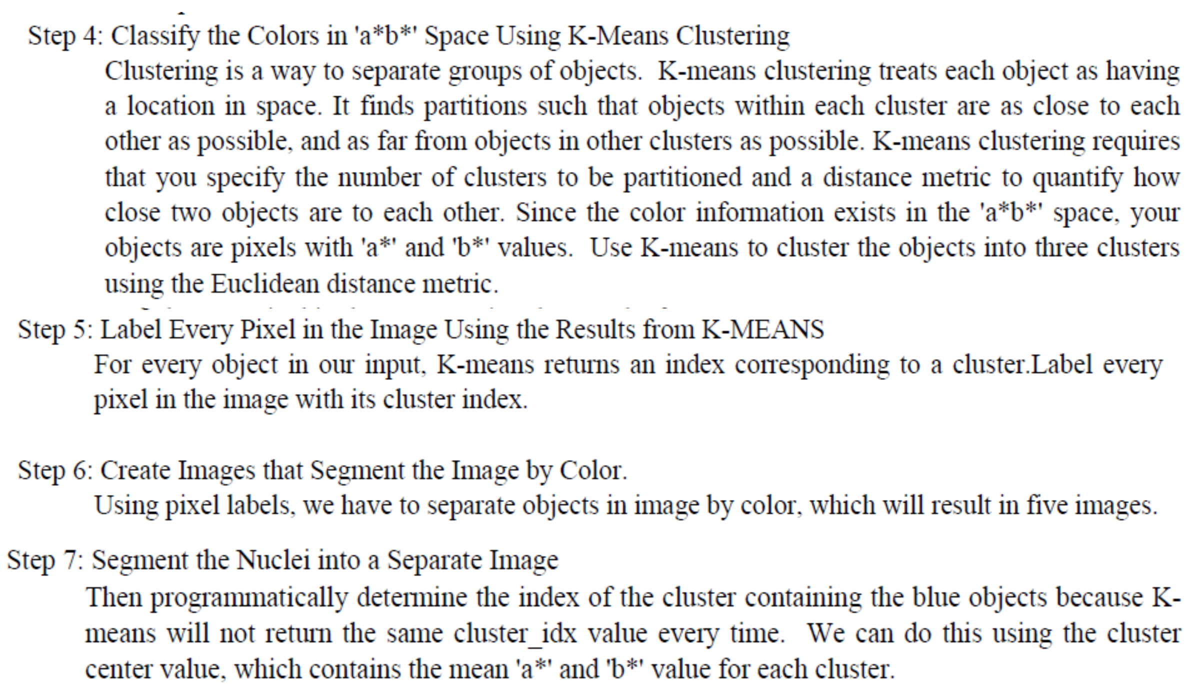
Step 1: Read the image

Read the image from mother source which is in .JPEG format, which is a fused image of part of Bhopal city of Madhya Pradesh, India with DWT fusion algorithm of Cartosat-1 and LISS-IV of Indian satellite IRS-P6 and IRS-1D.

Step 2: For colour separation of an image apply the Decorrelation stretching

Step 3: Convert Image from RGB Color Space to L*a*b* Color Space

How many colors do we see in the image if we ignore variations in brightness? There are three colors: white, blue, and pink. We can easily visually distinguish these colors from one another. The L*a*b* color space (also known as CIELAB or CIE L*a*b*) enables us to quantify these visual differences.The L*a*b* color space is derived from the CIE XYZ tristimulus values. The L*a*b* space consists of a luminosity layer 'L*', chromaticity-layer 'a*' indicating where color falls along the red-green axis, and chromaticity-layer 'b*' indicating where the color falls along the blue-yellow axis. All of the color information is in the 'a*' and 'b*' layers. We can measure the difference between two colors using the Euclidean distance metric. Convert the image to L*a*b*


Step 4: Classify the Colors in 'a*b*' Space Using K-Means Clustering

Clustering is a way to separate groups of objects. K-means clustering treats each object as having a location in space. It finds partitions such that objects within each cluster are as close to each other as possible, and as far from objects in other clusters as possible. K-means clustering requires that you specify the number of clusters to be partitioned and a distance metric to quantify how close two objects are to each other. Since the color information exists in the 'a*b*' space, your objects are pixels with 'a*' and 'b*' values. Use K-means to cluster the objects into three clusters using the Euclidean distance metric.

Step 5: Label Every Pixel in the Image Using the Results from K-MEANS

For every object in our input, K-means returns an index corresponding to a cluster.Label every pixel in the image with its cluster index.

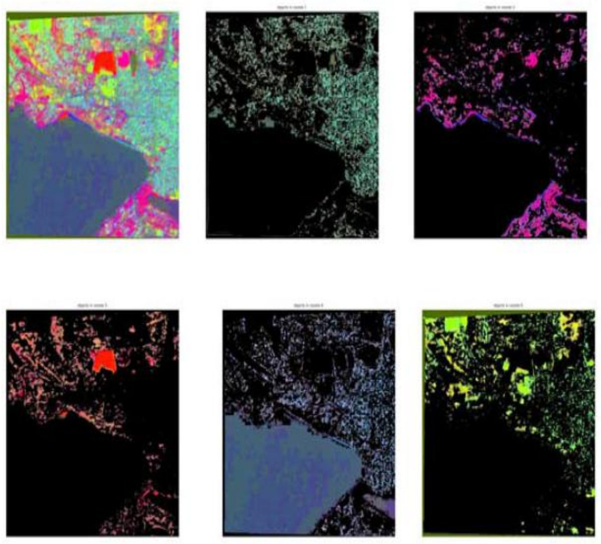
Step 6: Create Images that Segment the Image by Color.

Using pixel labels, we have to separate objects in image by color, which will result in five images.

Step 7: Segment the Nuclei into a Separate Image

Then programmatically determine the index of the cluster containing the blue objects because K-means will not return the same cluster_idx value every time. We can do this using the cluster center value, which contains the mean 'a*' and 'b*' value for each cluster.

Experimental Results:



[1] http://www.ijest.info/docs/IJEST10-02-10-117.pdf

FAQs

1.What is Color?

Ans: Color is the perceptual result of light in the visible region of the spectrum,having wavelengths in the region of 400 nm to 700 nm, incident upon the retina. Physical power (or *radiance*) is expressed in a *spectral power distribution* (SPD), often in 31 components each representing a 10 nm band. The human retina has three types of color photoreceptor *cone* cells, which respond to incident radiation with somewhat different spectral response curves. A fourth type of photoreceptor cell, the *rod* , is also present in the retina. Rods are effective only at extremely low light levels (colloquially, *night vision*), and although important for vision play no role in image reproduction.Because there are exactly three types of color photoreceptor, three numerical

components are necessary and sufficient to describe a color, providing that appropriate spectral weighting functions are used. This is the concern of the science of *colorimetry* . In 1931, the Commission Internationale
de L'Éclairage (CIE) adopted standard curves for a hypothetical *Standard Observer* . These curves specify how an SPD can be transformed into a set of three numbers that specifies a color.
2.What is Intensity?
Intensity is a measure over some interval of the electromagnetic spectrum of the flow of power that is radiated from, or incident on, a surface. Intensity is what I call a *linear-light* measure, expressed in units such as watts per square meter. The voltages presented to a CRT monitor control the intensities of the color components, but in a nonlinear manner. CRT voltages are not proportional to intensity.

3. What is Luinance?

Ans: *Brightness* is defined by the CIE as *the attribute of a visual sensationaccording to which an area appears to emit more or less light* . Because brightness perception is very complex, the CIE defined a more tractable quantity *luminance* which is radiant power weighted by a spectral sensitivity function that is characteristic of vision. The *luminous efficiency* of the Standard Observer is defined numerically, is everywhere positive, and peaks at about 555 nm. When an SPD is integrated using this curve as a weighting function, the result is *CIE luminance* denoted
Y. The magnitude of luminance is proportional to physical power. In that sense it is like intensity. But the spectral composition of luminance is related to the brightness sensitivity of human vision. Strictly speaking, luminance should be expressed in a unit such ascandelas per meter squared, but in practice it is often normalized to 1 or 100 units with respect to the luminance of a specified or implied *white reference*. For example, a studio broadcast monitor has a white reference whose luminance is about 80 cd•m-2, and *Y*= 1 refers to this value.

Review Questions

1. How can I characterize red, green and blue?
2. Why does offset printing use black ink in addition to CMY?
3. What is true color?
4. What is indexed color?

Refer : http://www.poynton.com/PDFs/ColorFAQ.pdf

Revised on:13/12/12

TITLE	Feature descriptor
PROBLEM STATEMENT/ DEFINITION	Analysis of different feature descriptor
OBJECTIVE	1. To understand concept of different feature spaces - using Fourier shape descriptors, experimenting with wavelet transform, template matching 2. To experiment above techniques with edge detection.
S/W PACKAGES AND HARDWARE APPARATUS USED	VC++/JAVA with OpenCV library, PC with minimum e configuration as Pentium IV 1.7 GHz. 128M.B RAM, 40 G.B HDD, 15"Color Monitor, Keyboard, Mouse
REFERENCES	R.O.Duda, “Pattern Classification” K.Fukunaga, “Statistical Pattern Recognition” O’reilley, “ Learning OpenCV”
STEPS	• Identification of application • Collection of data • Design of algorithm • Identification of openCV library function • Coding • Testing and analysis
INSTRUCTIONS FOR WRITING JOURNAL	Problem Definition Scope and limitation Theory OpenCv Library function used Algorithm Testing and Analysis Program Listing Output Conclusion

Aim: Analysis of different feature descriptor

Prerequisites:

- Feature Extraction techniques

Learning Objectives:

- To Study Different feature Extraction Techniques.
- Implement and Analysis Different feature Extraction Techniques on Given database

Learning Outcomes :

- Implementation of Identified feature extraction techniques on given image data
- Analysis of implemented techniques.

Theory/Concept:

The task of the feature extraction and selection methods is to obtain the most ***relevant information*** from the original data and represent that information in a ***lower dimensionality*** space.

When the input data to an algorithm is too large to be processed and it is suspected to be notoriously redundant (e.g. the same measurement in both feet and meters) then the input data will be transformed into a reduced representation set of features (also named features vector). Transforming the input data into the set of features is called *feature extraction*. If the features extracted are carefully chosen it is expected that the features set will extract the relevant information from the input data in order to perform the desired task using this reduced representation instead of the full size input. It can be used in the area of image processing which involves using algorithms to detect and isolate various desired portions or shapes (features) of a digitized image or video stream. It is particularly important in the area of optical character recognition

Low-level

- Edge detection
- Corner detection
- Blob detection
- Ridge detection
- Scale-invariant feature transform

Curvature

- Edge direction, changing intensity, autocorrelation.

Image motion

- Motion detection. Area based, differential approach. Optical flow.

Shape based

- Thresholding
- Blob extraction
- Template matching
- Hough transform

    - Lines
    - Circles/ellipses
    - Arbitrary shapes (generalized Hough transform)
    - Works with any parameterizable feature (class variables, cluster detection, etc.)

Flexible methods

- Deformable, parameterized shapes
- Active contours (snakes)

TITLE	PCA/ICA
PROBLEM STATEMENT/ DEFINITION	Experimentation of PCA and ICA, assuming sample feature set and Interpretation of ROC
OBJECTIVE	1. To use of ROC curves, experiments with PCA and ICA 2. To interpret ROC curves, experiments with PCA and ICA with sample feature sets.
S/W PACKAGES AND HARDWARE APPARATUS USED	VC++/JAVA with OpenCV library, PC with minimum e configuration as Pentium IV 1.7 GHz. 128M.B RAM, 40 G.B HDD, 15"Color Monitor, Keyboard, Mouse
REFERENCES	R.O.Duda, “Pattern Classification” K.Fukunaga, “Statistical Pattern Recognition” O’reilley, “ Learning OpenCV”
STEPS	• Identification of application • Collection of data • Design of algorithm • Identification of openCV library function • Coding • Testing and analysis
INSTRUCTIONS FOR WRITING JOURNAL	Problem Definition Scope and limitation Theory OpenCv Library function used Algorithm Testing and Analysis Program Listing Output Conclusion

Aim: Experimentation of PCA and ICA, assuming sample feature set and Interpretation of ROC

Prerequisites:

- Feature Extraction/Selection techniques
- Statistical concept

Learning Objectives:

- To Study PCA and ICA
- TO study Interpretation of ROC

Learning Outcomes :

- Experimentation of PCA and ICA, assuming sample feature set
- Analysis of implemented techniques using ROC

Theory/Concept:

Principal component analysis (PCA) is a mathematical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called **principal components**. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it be orthogonal to (i.e., uncorrelated with) the preceding components. Principal components are guaranteed to be independent only if the data set is jointly normally distributed. PCA is sensitive to the relative scaling of the original variables. Depending on the field of application, it is also named the discrete **Karhunen–Loève transform** (**KLT**), the **Hotelling transform** or **proper orthogonal decomposition** (**POD**).

Independent Component Analysis: [2]

Independent component analysis (**ICA**) is a computational method for separating a multivariate signal into additive subcomponents supposing the mutual statistical independence of the non-Gaussian source signals. It is a special case of blind source separation. ICA finds the independent components (aka factors, latent variables or sources) by maximizing the statistical independence of the estimated components. We may choose one of many ways to define independence, and this choice governs the form of the ICA algorithms. The two broadest definitions of independence for ICA are

1)Minimization of Mutual Information
2) Maximization of non-Gaussianity

The Minimization-of-Mutual information (MMI) family of ICA algorithms uses measures like Kullback-Leibler Divergence and maximum-entropy. The Non-Gaussianity family of ICA algorithms, motivated by the central limit theorem, uses kurtosis and negentropy.

Typical algorithms for ICA use centering, whitening (usually with the eigenvalue decomposition), and dimensionality reduction as preprocessing steps in order to simplify and reduce the complexity of the problem for the actual iterative algorithm. Whitening and dimension reduction can be achieved with principal component analysis or singular value decomposition. Whitening ensures that all dimensions are treated equally *a priori* before the algorithm is run. Algorithms for ICA include infomax, FastICA, and JADE, but there are many others.

In general, ICA cannot identify the actual number of source signals, a uniquely correct ordering of the source signals, nor the proper scaling (including sign) of the source signals.

ICA is important to blind signal separation and has many practical applications. It is closely related to (or even a special case of) the search for a factorial code of the data, i.e., a new vector-valued representation of each data vector such that it gets uniquely encoded by the resulting code vector (loss-free coding), but the code components are statistically independent.

Receiver Operating Characteristics:[3] Receiver operating characteristics (ROC) graphs are useful for organizing classifiers and visualizing their performance. ROC graphs are commonly used in medical decision making, and in recent years have been used increasingly in machine learning and data mining research. Although ROC graphs are apparently simple, there are some common misconceptions and pitfalls when using them in practice.

ROC graphs are conceptually simple, but there are some non-obvious complexities that arise when they are used in research. There are also common misconceptions and pitfalls when using them in practice. This article attempts to serve as a basic introduction to ROC graphs and as a guide for using them in research. The goal of this article is to advance general knowledge about ROC graphs so as to promote better evaluation practices in the field.
Given a classifier and an instance, there are four possibleoutcomes. If the instance is positive and it is classified aspositive, it is counted as a true positive; if it is classifiedas negative, it is counted as a false negative. If the instanceis negative and it is classified as negative, it is counted as atrue negative; if it is classified as positive, it is counted as afalse positive. Given a classifier and a set of instances (thetest set), a two-by-two confusion matrix (also called a contingency
table) can be constructed representing the dispositionsof the set of instances. This matrix forms the basis formany common metrics.Fig. 1 shows a confusion matrix and equations of severalcommon metrics that can be calculated from it. The numbersalong the major diagonal represent the correct decisionsmade, and the numbers of this diagonal representthe errors—the confusion—between the various classes.The true positive rate1 (also called hit rate and recall) of a classifier is estimated as

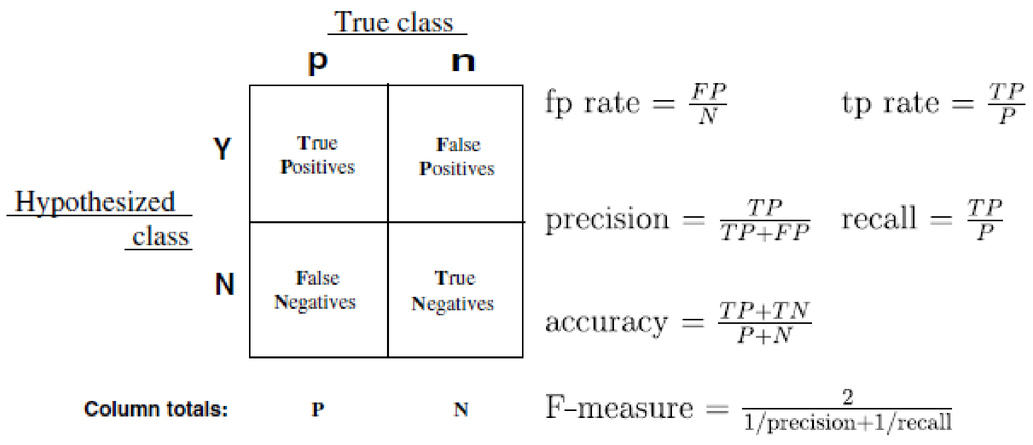


Fig. 1. Confusion matrix and common performance metrics calculated from it.

$$tp\ rate \approx \frac{\text{Positives correctly classified}}{\text{Total positives}}$$

The **false positive rate** (also called *false alarm rate*) of the classifier is

$$fp\ rate \approx \frac{\text{Negatives incorrectly classified}}{\text{Total negatives}}$$

Additional terms associated with ROC curves are

$$\begin{aligned} \text{sensitivity} &= \text{recall} \\ \text{specificity} &= \frac{\text{True negatives}}{\text{False positives} + \text{True negatives}} \\ &= 1 - fp\ rate \\ \text{positive predictive value} &= \text{precision} \end{aligned}$$

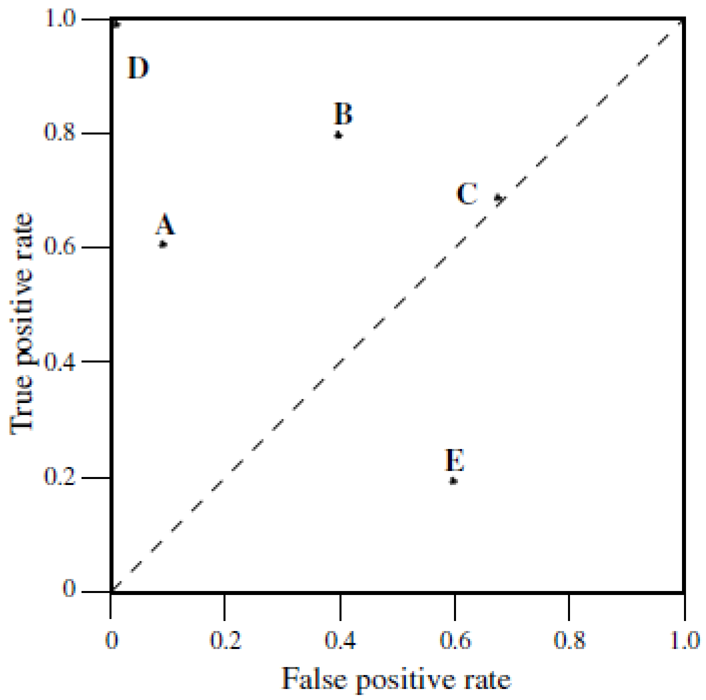


Fig. 2. A basic ROC graph showing five discrete classifiers.

Algorithm 1. Efficient method for generating ROC points
Inputs: L, the set of test examples; $f(i)$, the probabilistic classifier's estimate that example i is positive; P and N, the number of positive and negative examples.
Outputs: R, a list of ROC points increasing by *fp rate*.
Require: $P > 0$ and $N > 0$

```
 1: $L_{\text{sorted}} \leftarrow L$ sorted decreasing by $f$ scores
 2: $FP \leftarrow TP \leftarrow 0$
 3: $R \leftarrow \langle\rangle$
 4: $f_{\text{prev}} \leftarrow -\infty$
 5: $i \leftarrow 1$
 6: **while** $i \leqslant |L_{\text{sorted}}|$ **do**
 7:     **if** $f(i) \neq f_{\text{prev}}$ **then**
 8:         push $\left(\frac{FP}{N}, \frac{TP}{P}\right)$ onto $R$
 9:         $f_{\text{prev}} \leftarrow f(i)$
10:     **end if**
11:     **if** $L_{\text{sorted}}[i]$ is a positive example **then**
12:         $TP \leftarrow TP + 1$
13:     **else**        /* $i$ is a negative example */
14:         $FP \leftarrow FP + 1$
15:     **end if**
16:     $i \leftarrow i + 1$
17: **end while**
18: push $\left(\frac{FP}{N}, \frac{TP}{P}\right)$ onto $R$   /* This is (1,1) */
19: **end**
```

Algorithm 2. Calculating the area under an ROC curve
Inputs: L, the set of test examples; $f(i)$, the probabilistic classifier's estimate that example i is positive; P and N, the number of positive and negative examples.
Outputs: A, the area under the ROC curve.
Require: $P > 0$ and $N > 0$

```
 1: $L_{\text{sorted}} \leftarrow L$ sorted decreasing by $f$ scores
 2: $FP \leftarrow TP \leftarrow 0$
 3: $FP_{\text{prev}} \leftarrow TP_{\text{prev}} \leftarrow 0$
 4: $A \leftarrow 0$
 5: $f_{\text{prev}} \leftarrow -\infty$
 6: $i \leftarrow 1$
 7: while $i \leqslant |L_{\text{sorted}}|$ do
 8:    if $f(i) \neq f_{\text{prev}}$ then
 9:       $A \leftarrow A + \text{TRAPEZOID\_AREA}(FP, FP_{\text{prev}},$
              $TP, TP_{\text{prev}})$
10:       $f_{\text{prev}} \leftarrow f(i)$
11:       $FP_{\text{prev}} \leftarrow FP$
12:       $TP_{\text{prev}} \leftarrow TP$
13:    end if
14:    if $i$ is a positive example then
15:       $TP \leftarrow TP + 1$
16:    else      /* $i$ is a negative example */
17:       $FP \leftarrow FP + 1$
18:    end if
19:    $i \leftarrow i + 1$
20: end while
21: $A \leftarrow A + \text{TRAPEZOID\_AREA}(N, FP_{\text{prev}}, N, TP_{\text{prev}})$
22: $A \leftarrow A/(P \times N)$   /* scale from $P \times N$ onto the unit
    square */
23: end
```

```
 1: function TRAPEZOID_AREA($X1, X2, Y1, Y2$)
 2: $Base \leftarrow |X1 - X2|$
 3: $Height_{\text{avg}} \leftarrow (Y1 + Y2)/2$
 4: return $Base \times Height_{\text{avg}}$
 5: end function
```

FAQS:
1.What is the Primary reasons for using PCA
Ans:

There are two primary reasons for using PCA:

Data Reduction PCA is most commonly used to condense the information contained in a large number of original variables into a smaller set of new composite variables or dimensions, at the same time ensuring a minimum loss of information.

Interpretation
PCA can be used to discover important features of a large data set. It often reveals relationships that were previously unsuspected, thereby allowing interpretations of the data that may not ordinarily result from examination of the data. PCA is typically used as an intermediate step in data analysis when the number of input variables is otherwise too large to perform useful analysis.

2. What is **t**he area under the ROC curve ?

The area under a ROC curve quantifies the overall ability of the test to discriminate between those individuals with the disease and those without the disease.

References:
[1] www.ce.yildiz.edu.tr/personal/songul/file/.../**principal_components**

[2] ftp://ftp.rta.nato.int/pubfulltext/RTO/MP/RTO-MP-SET-080/MP-SET-080-37.pdf

[3] https://ccrma.stanford.edu/workshops/mir2009/references/ROCintro.pdf

[4] http://gim.unmc.edu/dxtests/roc2.htm