Pattern Recognition and Machine Learning

James L. Crowley

ENSIMAG 3 - MMIS
Lesson 1

Fall Semester 2018
10 October 2018

Learning and Evaluation for Pattern Recognition

Outline

Notation

x_d	A feature. An observed or measured value.
$\vec{X}$	A vector of features. An observation.
D	The number of dimensions for the vector $\vec{X}$
$\{C_k\}$	A set of K classes (or class labels).
K	Number of classes
$\vec{X} \in C_k$	Statement that the observation $\vec{X}$ is a member of class C_k
$\hat{C}_k$	An estimated class label
	For a 2 class detection problem (K=2), $C_k \in \{P,N\}$
$R(\vec{X})$	A recognition function
$\hat{C}_k \leftarrow R(\vec{X})$	A recognition function that predicts $\hat{C}_k$ from $\vec{X}$
$\{\vec{X}_m\}$	Training data for learning.
M	The number of training samples.
$y(\vec{X}_m)$	An annotation (or ground truth) function for $\{\vec{X}_m\}$: $y(\vec{X}_m) \in \{P,N\}$
$g(\vec{X}_m)$	Discriminant function. $0 \le g(\vec{X}_m) \le 1$
$X(i,j)$	An RGB image of size RxC pixels, 8 bits per color
$P(i,j)$	A probability image of size RxC pixels. Each pixel contains the probability (or likelihood) that the pixel (i,j) is a part of a face.
$\{X_n(i,j)\}$	A set of N images for training. $\vec{X}_m = X_n(i,j)$ where $m=n \cdot i \cdot j$
N	The number of training images.
$y(X_n(i,j))$	A ground-truth function that tells if pixel (i,j) of image n is part of a face.
M_T	The number of training pixels in the target class
$h(\vec{X})$	A multidimensional histogram of integer features $\vec{X}$
Q	The number of discrete values for each dimension (quantization) of $h(\vec{X})$
V	The number of cells in the histogram $h(\vec{X})$
$W_n(u,v)$	A rectangular window at (c_i, c_j) and size (width, height)
	spanning from (l, t) to (b, r)

1. Organization of the course

In this course, we will use face detection as a running example to illustrate different learning techniques. We will implement and experimentally evaluate three different techniques for face detection in images.

Lab 1: Face detection using pixel level skin color detection.
Lab 2: Face detection using the Viola Jones cascade detector
Lab 3: Face detection using multilayer neural networks.

Lab exercises will be programmed, evaluated and reported by teams of 3 students.
Each team will make an oral presentation on one of the 3 labs.

Labs will be performed using the OpenCV environment running under Python.
Groups will be encouraged to be creative in implementing, evaluating and reporting the labs. The labs may use code found downloaded from the internet PROVIDED that you document the origin of the code. The primary task is performance evaluation.

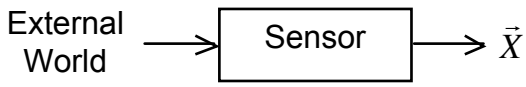
Grades are determined 50% from the labs and 50% from a final exam.
The lab grade is the average from the grades of 3 written reports of the team members plus the oral reports. Written reports will be due 1 week after the oral reports.

Team compositions are to be finalized at the second lecture.

2. The Pattern Recognition Problem

Pattern Recognition (or classification) is the process of assigning observations to categories. The observation is sometimes called an "entity" or a "sample" depending on the context and domain.

Observations are produced by some form of sensor. A sensor is a transducer that transforms physical phenomena into digital measurements.
These measurements are classically called "Features".



Features may be Boolean, natural numbers, integers, real numbers or symbolic labels.

In most interesting problems, the sensor provides a vector of *D* features, $\vec{X}$.

$$\vec{X} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{pmatrix}$$

Our problem is to build a function, called a recognizer or classifier, $\mathrm{R}(\vec{X})$, that maps the observation, $\vec{X}$ into a statement that the observation belongs to a class $\hat{C}_k$ from a set of K possible classes. $R(\vec{X}) \rightarrow \hat{C}_k$

$$\vec{X} \rightarrow \boxed{R(\vec{X})} \rightarrow \hat{C}_k$$

In most classic techniques, the class $\hat{C}_k$ is from a set of K known classes $\{C_k\}$.

$\{C_k\}$ is a generally a closed set. Almost all current classification techniques require the number of classes, K, to be fixed.
An interesting research problem is how to design classification algorithms that allow $\{C_k\}$ to grow with experience.

Detection functions are a special case of recognition.
For a detection function, there are 2 classes (K=2). $C_k \in \{P, N\}$

Class 1 *(C_1)* is a positive detection P.
Class 2 *(C_2)* is a negative detection, N.

2.1 Discriminant and Decision Functions

The classification function $R(\vec{X})$ can typically be decomposed into two parts:

$$\hat{C}_k \leftarrow R(\vec{X}) = d\left(\vec{g}\left(\vec{X}\right)\right)$$

where $\vec{g}\left(\vec{X}\right)$ is a discriminant function and $d\left(\vec{g}\left(\vec{X}\right)\right)$ is a decision function.

$\vec{g}\left(\vec{X}\right)$: A discriminant function that transforms: $\vec{X} \rightarrow R^N$

The discriminant function is typically learned from the data.

$d\left(\vec{g}\left(\vec{X}\right)\right)$: A non-linear decision function $R^N \rightarrow \hat{C}_k \in \{C_k\}$

The decision function is chosen by the system designer.

Many modern techniques interleave several layers of discrimination and decision.

$$R(\vec{X}) = d\left(\vec{g}\left(d\left(\vec{g}\left(...\right)\right)\right)\right)$$

2.2 Machine Learning

Machine learning explores the study and construction of algorithms that can learn from and make predictions about data.

Learning for Pattern Recognition is one of many forms of machine learning. Over the last 50 years, many forms of machine learning have been developed for many different applications. Machine learning techniques have been demonstrated from Music Synthesis, Speech synthesis, Image synthesis and Robot control (manipulation, walking, expressing emotions) among other areas.

For pattern recognition, the common approach is to a set of "training data" to estimate the discriminant function $\vec{g}\left(\vec{X}\right)$.

The danger with supervised learning is that the model may not generalize to data outside the training set. The quality of the recognizer depends on the degree to which the training data $\{\vec{X}_m\}$ represents the range of variations of real data.

A variety of algorithms have been developed, each with its own advantages and disadvantages.

Supervised Learning

Most classical methods for learning a recognition function, learn from a set of labeled training data, composed of *M* independent examples, $\{\vec{X}_m\}$ for which we know the true class $\{y_m\}$.

The set $\{\vec{X}_m\}$, $\{y_m\}$ is called the training data.

Having the true class $\{y_m\}$ makes it much easier to estimate the functions $g_k(\vec{X})$
Most of classic techniques for machine learning use supervised learning.

Unsupervised Learning

Unsupervised Learning techniques learn the recognition function without a labeled training set. Such methods typically require a much larger sample of data for learning. We will see how to use EM and K-means for unsupervised learning.

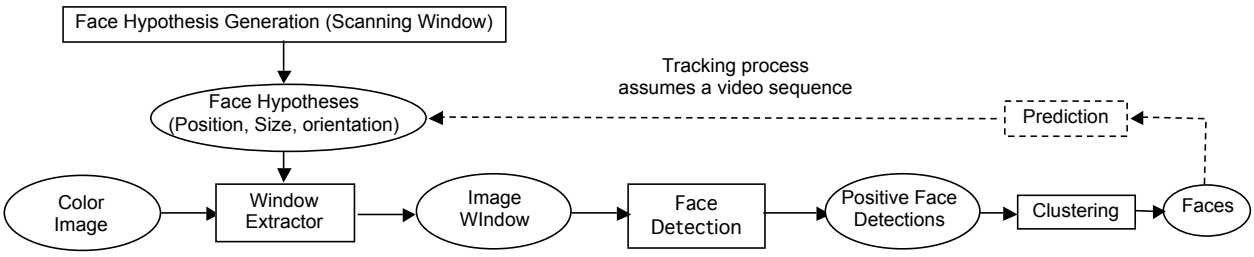
Semi-Supervised Learning.

A number of hybrid algorithms exist that initiate learning from a labeled training set and then extend the learning with unlabeled data.

3. Face Detection

Faces can occur at many different positions, orientations and sizes (scales) in an image. A common approach is to train a detection function with a standard size and orientation of faces. A sliding window process is then designed to extract windows from the face for a range of positions, orientations and sizes (scales). The contents of each window are transformed (normalized) to the size and orientation used by the recognizer. This transformation is easily performed by the "texture mapping" function found in OpenCV. The normalized window is then processed by the recognition function.

Recognition typically returns a "likelihood" estimate for a face at many adjacent positions, orientations and sizes. It is necessary to cluster these adjacent detections to determine the best position for a single face. This can be done using the center of gravity of the likelihoods.

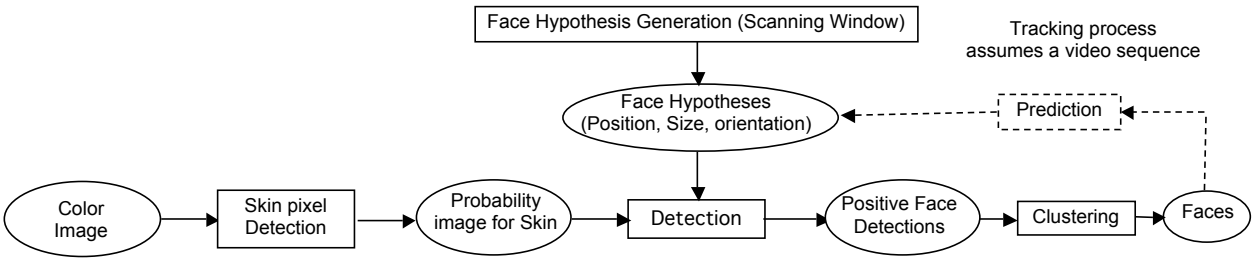


The same process can be used for tracking previous detected faces in a video sequence. In this case, face detection is limited to a small range of positions, sizes and orientations, estimated from the detection from the previous image.

3.1 Face Detection using skin color

Skin color can be used to construct a simple detector for skin pixels in images. Color skin pixels can then be used to detect and track “blobs” that represent faces, hands and other skin colored regions in images. Vertically oriented blobs of a certain size are assumed to be faces. This is an important source of error.

The detector works by first computing the probability that each pixel contains skin. A sliding window (Region of Interest or ROI) is then scanned over the image. At each position, a weighted sum of probabilities is determined. Regions for which the weighted sum is above threshold are detected as faces. Adjacent face detections are clustered to form a single face detection.



Algorithm:
1) Compute probability of skin at each pixel.
2) Sum the probability within a window at a position, size and orientation.
(this can be a weighted sum.) Determine if detection is Sum is above threshold.
3) Cluster adjacent detections, for example with center of gravity.
4) Estimate precise face position, size and orientation from clusters of detections.

3.2 Detecting skin pixels with color histograms.

In the following, assume that we have a color image, where each pixel *(i,j)* is a color vector, $X(i,j)$, composed of 3 integers between 0 and 255 representing Red, Green and Blue.

$$X(i,j) = \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Each color is represented by value between 0 and 255 (8 bits).

Suppose that we have N color images of size CxR pixels, $X_n(i,j)$.
This gives a total of $M = C \times R \times N$ pixels.
Call this set of pixels $\vec{X}_m = X_n(i,j)$ where $m = i \cdot j \cdot n$

Suppose that we have a ground truth function $y(X_m)$ that tells us whether each pixel is target (P or Positive) or not target (N or Negative). A subset of M_T pixels that belong to a target class, T.

We allocate two tables $h(\vec{X})$ and $h_T(\vec{X})$ and use these to construct two histograms.
we can also write this as $\forall_m h(\vec{X}_m) = h(\vec{X}_m) + 1$

and $\forall_m y(\vec{X}_m) = P : \; h_T(\vec{X}_m) = h_T(\vec{X}_m) + 1 \; ; \; M_T \leftarrow M_T + 1$

Thus for any color pixel, $\vec{X}$, we have two probabilities:

$$P(\vec{X}) = \frac{1}{M} h(\vec{X}) \quad \text{and} \quad P(\vec{X} \mid T) = \frac{1}{M_T} h_T(\vec{X})$$

Bayes rule tells us that we can estimate the probability that a pixel belongs to target class (Skin) given its color, $\vec{X}$ as:

$$P(Skin \mid \vec{X}) = \frac{P(\vec{X} \mid \text{Skin}) P(\text{Skin})}{P(\vec{X})}$$

P(Skin) is the probability that a pixel belongs to the target class.
This can be estimated from the training data by:

$$P(Skin) = \frac{M_T}{M}$$

From this we can show that the probability that a pixel is skin, is simply the ratio of the two tables.

$$P(Skin \mid \vec{X}) = \frac{P(\vec{X} \mid Skin) P(Skin)}{P(\vec{X})} = \frac{\frac{1}{M_T} h_t(\vec{X}) \cdot \frac{M_t}{M}}{\frac{1}{M} h(\vec{X})} = \frac{h_T(\vec{X})}{h(\vec{X})}$$

We can use this to compute a lookup table $L_{Skin}(\vec{X}) = \frac{h_T(\vec{X})}{h(\vec{X})}$

Note that if $h(\vec{X}) = 0$ then $h_{Skin}(\vec{X}) = 0 = 0$ because $h_{Skin}(\vec{X})$ is a subset of $h(\vec{X})$.
we will need to test this case to avoid divide by 0.

If we ASSUME that a new image, *X(i,j)*, has similar illumination and color composition then we can use this technique to assign a probability to each pixel by table lookup. The result is an image in which each pixel is a probability *P(i,j)* that the pixel *(i,j)* belongs to class skin.

$$P(i,j) = L_{Skin}(\vec{X}(i,j))$$

Details:

1) alternative color codings may provide better recognition.
2) Different color quantizations can provide better recogntion.

This will be discussed in next weeks lecture.
You will be asked to test this in the first exercise.

3.4 Detecting skin blobs.

We will assume faces are vertical skin blobs.
Essentially we assume that any region of a certain width and height that contains many skin colored pixels is a face.

This is a strong assumption that will be an important source of errors.

Hypotheses for the presence of a skin blob at a particular position and size can be tested as the sum of skin probabilities with a rectangle at a position (c_i, c_j) and size (width, height). This rectangle is typically called a "region of interest" or ROI and represented as a rectangular window defined as a vector of four corners: (top, left, bottom, right) or *(t,b,l,r)*.

Let us define a face hypothesis as a ROI: $\vec{W}_n = \begin{pmatrix} t \\ l \\ b \\ r \end{pmatrix}$

The likelihood of a skin blob at the ROI is the average probability.

$$L_{face}(\vec{W}_n) = \frac{1}{(t-b)(l-r)} \sum_{i=l}^{r} \sum_{j=t}^{b} P(i,j)$$

If $L_{face}(\vec{W}_n) \geq$ Threshold then Face (Positive Detection or P) else NOT Face (Negative Detection or N)

We can bias the detection by adding a bias term B.

If $L_{face}(\vec{W}_n) + B \geq$ Threshold then P else NOT P

We can improve detection by weighting the probabilities with a Gaussian or face shaped mask. This is described next week by Nachwa. .

Scanning window detector.

A scanning window detector systematically tests hypotheses over a range of positions and sizes. This can be made more efficient by a form of hierarchical search.
This is discussed in next weeks lecture.

Clustering adjacent detections:
When a face is present, it will be detected at multiple adjacent positions and sizes. The best face position can be obtained by "grouping" adjacent detections.

This is discussed in next weeks lecture.

3.5 Test Data

For lab exercises, you can use the FDDB (Face Detection Data Set and Benchmark) data base maintained at UMASS: http://vis-www.cs.umass.edu/fddb/

This data-base was constructed for face detection and not for skin detection.
Face regions have been hand-labeled as both boxes and ellipses.

All images are RGB with each pixel containing 3 colors: Red, Green and Blue.

We will use the ellipses as ground truth for skin regions. A typical image with and annotated face regions as an ellipse looks like.

Any pixel inside the ellipse can be considered skin and included in the skin color histogram.

Note that there are skin pixels that are NOT in the ellipse (hand, ears, neck etc), and there are non-skin pixels that ARE in the face (hair, teeth, etc). This will lead to minor errors in the detection. Your job will be to measure the impact of these errors in building a face detector using detection of skin pixels.

Note that faces appear with an elliptical form, with major axis in the vertical direction. Annotations of face regions in FDDB are represented as an elliptical

region, denoted by a 6-tuple (r_a, r_b, θ, c_x, c_y, 1) where r_a and r_b refer to the half-length of the major and minor axes, θ is the angle of the major axis with the horizontal axis, and c_x and c_y are the column and row image coordinates of the center of this ellipse.

Ellipse Data:
2002/07/24/big/img_82
1
59.268600 35.142400 1.502079 149.366900 59.365500 1

the standard form of an ellipse with a major axis along the horizontal (x) axis is:

$$\frac{(x-c_x)^2}{r_a^2}+\frac{(y-c_y)^2}{r_b^2}=1$$

for any pixel x,y inside the ellipse, $\frac{(x-c_x)^2}{r_a^2}+\frac{(y-c_y)^2}{r_b^2}<1$

For a hypothesis of a face $\vec{X}=\begin{pmatrix} c_x \\ c_y \\ r_a \\ r_b \\ \theta \end{pmatrix}$ can define a ground truth function as $y(\vec{X}_m)$

$$y(\vec{X}_m)=if(\frac{(x-c_x)^2}{r_a^2}+\frac{(y-c_y)^2}{r_b^2}\le 1) \text{ then P else N.}$$

If it is necessary to rotate the face to an angle θ we can use:

$$\frac{\left((x-c_x)\cos(\theta)+(y-c_y)\sin(\theta)\right)^2}{r_a^2}+\frac{\left((x-c_x)\sin(\theta)+(y-c_y)\cos(\theta)\right)^2}{r_b^2}=1$$

Face hypotheses can be limited to a single size or tested over a range of sizes. We can use the major and minor ellipses to define the range of height and width over which we need to find faces.

3.6 Training and Validation

With Supervised Learning, we estimate discriminant functions from a "labeled" training set of observations $\{\vec{X}_m\}$ where each sample observation is labeled with an indicator variable $\{y_m\}$ or an indicator function $y_m\left(\vec{X}_m\right)$.

$y_m =$ P or Positive for examples of the target pattern (class k=1)
$y_m =$ N or Negative for all other examples (class k=2)

The detection function is learned from a subset of the training data then tested with a different subset of the training data

NEVER TRAIN and TEST with the SAME DATA !

A typical approach is to use cross validation (also known as rotation estimation) for training and for evaluation. Cross validation partitions the training data into N folds (or complementary subsets). A subset of the folds are used to train the classifier, and the result is tested on the other folds. A taxonomy of common techniques include:

- Exhaustive cross-validation
  - Leave p-out cross-validation
  - Leave one-out cross-validation

- Non-exhaustive cross-validation
  - k-fold cross-validation
  - 2-fold cross-validation
  - Repeated sub-sampling validation

4. Performance Evaluation for Pattern Detectors

4.1 True and False Positives and Negatives,

A pattern detector is a classifier with K=2.

Class k=1: The target pattern, also known as P or positive

Class k=2: Everything else, also known as N or negative.

Pattern detectors are used in computer vision, for example to detect faces, road signs, publicity logos, or other patterns of interest. They are also used in signal communications, data mining and many other domains.

Assume that we have M training samples, $\{\vec{X}_m\}$ along with a function $y(\vec{X}_m)$ that tells if a pixel is in the target class P or N. For face detection, this will be N images $X_n(i,j)$ where $\vec{X}_m = X_n(i,j)$ along with a function, $y(X_n(i,j))$ that tells is a pixel belongs to a face.

The pattern detector is learned as a detection function $g(\vec{X})$ followed by a decision rule, *d()*.

For example, the decision rule can be : if $g(\vec{X}) + B \geq 0.5$ then P else N

Observations for which $g(\vec{X}) + B > 0.5$ are estimated to be members of the target class. These are POSITIVE or P.

Observations for which $g(\vec{X}) + B \leq 0.5$ are estimated to be members of the background. These are NEGATIVE or N.

We can encode the decision function to define our detection function $R(\vec{X}_m)$ as

$$R(\vec{X}) = d(g(\vec{X}) + B) = \begin{cases} P & \text{if } g(\vec{X}) + B \geq 0.5 \\ N & \text{if } g(\vec{X}) + B < 0.5 \end{cases}$$

For training we need ground truth (annotation). For each training sample the annotation or ground truth tells us the real class y_m

$$y_m = \begin{cases} P & \vec{X}_m \in \text{Target - Class} \\ \text{N} & \text{otherwise} \end{cases}$$

The Classification can be TRUE or FALSE.

if $R(\vec{X}_m) = y_m$ then T else F

This gives

$R(\vec{X}_m) = y_m$ AND $R(\vec{X}_m) = P$ is a TRUE POSITIVE or TP
$R(\vec{X}_m) \neq y_m$ AND $R(\vec{X}_m) = P$ is a FALSE POSITIVE or FP
$R(\vec{X}_m) \neq y_m$ AND $R(\vec{X}_m) = N$ is a FALSE NEGATIVE or FN
$R(\vec{X}_m) = y_m$ AND $R(\vec{X}_m) = N$ is a TRUE NEGATIVE or TN

To better understand the detector we need a tool to explore the trade-off between making false detections (false positives) and missed detections (false negatives). The Receiver Operating Characteristic (ROC) provides such a tool.

4.2 ROC Curves

Two-class classifiers have long been used for signal detection problems in communications and have been used to demonstrate optimality for signal detection methods. The quality metric that is used is the Receiver Operating Characteristic (ROC) curve. This curve can be used to describe or compare any method for signal or pattern detection.

The ROC curve is generated by adding a variable Bias term to a discriminant function.

$$R(\vec{X}) = d(g(\vec{X}) + B)$$

and plotting the rate of true positive detection vs false positive detection where $R(\vec{X}_m)$ is the classifier as in lesson 1. As the bias term, B, is swept through a range of values, it changes the ratio of true positive detection to false positives.

For a ratio of histograms, $g(\vec{X}_m)$ is a probability ranging from 0 to 1.
The bias term, B, can act as an adjustable gain that sets the sensitivity of the detector.
The bias term allows us to trade False Positives for False Negatives.

B can range from less than -0.5 to more than $+0.5$.
When $B \leq -0.5$ all detections will be Negative.
When $B > +0.5$ all detections will be Positive.
Between -0.5 and $+0.5$ $R(\vec{X})$ will give a mix of TP, TN, FP and FN.

The resulting curve is called a Receiver Operating Characteristics (ROC) curve.
The ROC plots True Positive Rate (TPR) against False Positive Rate (FNR) as a function of B for the training data $\{\vec{X}_m\}$, $\{y_m\}$.

For each training sample, the detection as either Positive (P) or Negative (N)

IF $g(\vec{X}_m)+B > 0.5$ THEN P else N

The detection can be TRUE (T) or FALSE (F) depending on the indicator variable y_m

IF $y_m = R(\vec{X}_m)$ THEN T else F

Combining these two values, any detection can be a True Positive (TP), False Positive (FP), True Negative (TN) or False Negative (FN).

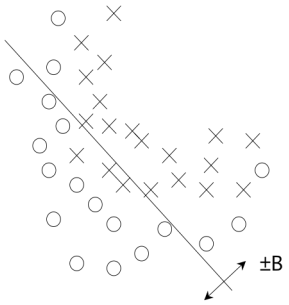
For the M samples of the training data $\{\vec{X}_m\}$, $\{y_m\}$ we can define:
#P as the number of Positives,
#N as the number of Negatives,
#T as the number of True and
#F as the number of False,

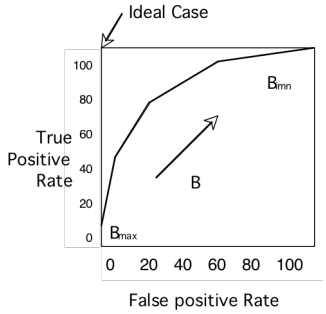
From this we can define:
#TP as the number of True Positives,
#FP as the number of False Positives,
#TN as the number of True Negative,
#FN as the number of False Negatives.

Note that #P = #TP + #FN
And #N = #FP+ #TN
The True Positive Rate (TPR) is $TPR = \frac{\#TP}{\#P} = \frac{\#TP}{\#TP+\#FN}$

The False Positive Rate (FPR) is $FPR = \frac{\#FP}{\#N} = \frac{\#FP}{\#FP+\#TN}$

The ROC plots the TPR against the FPR as a bias B is swept through a range of values.





When B is less than –0.5, all the samples are detected as N, and both the TPR and FPR are 0. As B increases both the TPR and FPR increase. Normally TPR should rise monotonically with FPR. If TPR and FPR are equal, then the detector is no better than chance.

The closer the curve approaches the upper left corner, the better the detector.

		$y_m = R(\vec{X}_m)$	
$d(g(\vec{X}_m)+B > 0.5)$		T	F
	P	True Positive (TP)	False Positive (FP)
	N	True Negative (TN)	False Negative (FN)

4.3 Precision and Recall

Precision, also called Positive Predictive Value (PPV), is the fraction of retrieved instances that are relevant to the problem.

$$PP = \frac{TP}{TP + FP}$$

A perfect precision score (PPV=1.0) means that every result retrieved by a search was relevant, but says nothing about whether all relevant documents were retrieved.

Recall, also known as sensitivity (S), hit rate, and True Positive Rate (TPR) is the fraction of relevant instances that are retrieved.

$$S = TPR = \frac{TP}{T} = \frac{TP}{TP + FN}$$

A perfect recall score (TPR=1.0) means that all relevant documents were retrieved by the search, but says nothing about how many irrelevant documents were also retrieved.

Both precision and recall are therefore based on an understanding and measure of relevance. In our case, "relevance" corresponds to "True".
Precision answers the question "How many of the Positive Elements are True ?"
Recall answers the question "How many of the True elements are Positive"?

In many domains, there is an inverse relationship between precision and recall. It is possible to increase one at the cost of reducing the other.

4.4 F-Measure

The F-measures combine precision and recall into a single value. The F measures measure the effectiveness of retrieval with respect to a user who attaches 2 times as much importance to recall as precision.

The F_1 score weights recall higher than precision.

4.5 F_1 Score:

$$F_1 = \frac{2TP}{2TP + FP + FN}$$

The F1 score is the harmonic mean of precision and sensitivity.
This is the geometric mean divided by the arithmetic mean.

4.6 Accuracy

Accuracy is the fraction of test cases that are correctly classified (T).

$$ACC = \frac{T}{M} = \frac{TP + TN}{M}$$

where M is the quantity of test data.

Note that the terms Accuracy and Precision have a very different meaning in Measurement theory. In measurement theory, accuracy is the average distance from a true value, while precision is a measure of the reproducibility for the measurement.

4.7 Matthews Correlation Coefficient

The Matthews correlation coefficient is a measure of the quality of binary (two-class) classifications. This measure was proposed by the biochemist Brian W. Matthews in 1975.

MCC takes into account true and false positives and negatives and is generally regarded as a balanced measure that can be used even if the classes are of very different sizes.

The MCC is in essence a correlation coefficient between the observed and predicted binary classifications

MCC results a value between +1 and -1, where +1 represents a perfect prediction, 0 no better than random prediction and −1 indicates total disagreement between prediction and observation.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

The original formula given by matthews was:

M = Total quanitity of test data: $M = TN + TP + FN + FP$

$$S = \frac{TP + FN}{M} \qquad P = \frac{TP + FP}{M}$$

$$MCC = \frac{TP/M - S \cdot P}{\sqrt{PS(1-S)(1-P)}}$$