Pattern Recognition and Machine Learning

James L. Crowley

ENSIMAG 3 - MMIS | Fall Semester 2019
Lesson 3 | 23 October 2019

Performance Evaluation for Machine Learning

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. The Pattern Detection Problem

Pattern Recognition (or classification) is the process of assigning observations to categories. 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".

$$\begin{tikzcd}\text{External World} \arrow[r] & \boxed{\text{Sensor}} \arrow[r] & \vec{X}\end{tikzcd}$$

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 pattern Detector. This is a special case of a 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. $R(\vec{X}) \to \hat{C}_k$

$$\begin{tikzcd}\vec{X} \arrow[r] & \boxed{R(\vec{X})} \arrow[r] & \hat{C}_k\end{tikzcd}$$

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.

1.1 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: $\vec{X} \rightarrow R^K$

The discriminant function is typically learned from the data.

$d\left(\vec{g}\left(\vec{X}\right)\right)$: A non-linear decision function chosen by the system designer.

$R^K \rightarrow \hat{C}_k \in \{C_k\}$

1.2 Supervised Learning

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

Most of classic techniques for machine learning use only supervised learning. Some modern techniques permit semi-supervised or unsupervised learning after an initial training with supervised learning.

The model 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

2. Performance Evaluation for Pattern Detectors

2.1 Pattern Detectors

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 speech recognition, acoustic sensing, signal communications, data mining and many other domains.

The pattern detector is learned as a detection function $g(\vec{X})$ followed by a decision rule, *d()*. For K=2 this can be reduced to a single function, as

$g_1(\vec{X}) \geq g_2(\vec{X})$ is equivalent to $g(\vec{X}) = g_1(\vec{X}) - g_2(\vec{X}) \geq 0$

Note that a "threshold" value, B, other than 0 can be used. This is equivalent to "biasing" the detector.

The detection function is learned from a set of training data composed of *M* sample observations $\{\vec{X}_m\}$ where each sample observation is labeled with an indicator variable $\{y_m\}$

$y_m =$ P or Positive for examples of the target pattern (class k=1)

$y_m =$ N or Negative for all other examples (class k=2)

Observations for which $g(\vec{X}) + B > 0$ are estimated to be members of the target class. This will be called POSITIVE or P.

Observations for which $g(\vec{X}) + B \leq 0$ are estimated to be members of the background. This will be called NEGATIVE or N.

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

$$R(\vec{X}) = d(g(\vec{X})) = \begin{cases} P & \text{if } g(\vec{X}) + B \geq 0 \\ N & \text{if } g(\vec{X}) + B < 0 \end{cases}$$

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

2.2 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.

When $g_1(\vec{X})$ and $g_2(\vec{X})$ are probabilities,
then $g(\vec{X}) = g_1(\vec{X}) - g_2(\vec{X})$ will range from –1 to +1.
When $B < -1$ all detections will be Negative.

When $B > +1$ all detections will be Positive.
Between -1 and $+1$, $R(\vec{X})$ will give a mix of TP, TN, FP and FN.

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.

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 $\{\vec{X}_m\}$, $\{y_m\}$.

2.3 True Positives and False Positives

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

IF $g(\vec{X}_m)+B > 0$ 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).

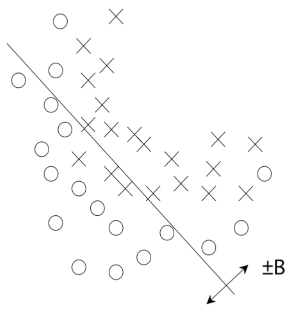
For the M samples of the training data $\{\vec{X}_m\}$, $\{y_m\}$ we can define:
#P as the number of Positives in the training data.
#N as the number of Negatives in the training data.
#T as the number of training samples correctly labeled by the detector.
#F as the number of training samples incorrectly labeled by the detector.
From this we can define:
#TP as the number of training samples correctly labeled as Positive
#FP as the number of training samples incorrectly labeled as Positive
#TN as the number of training samples correctly labeled as Negative
#FN as the number of training samples incorrectly labeled as Negative

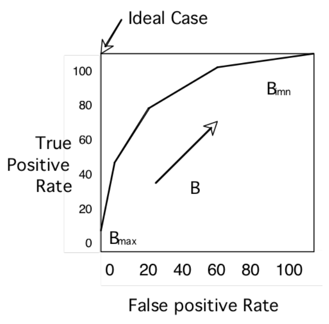
Note that #P = #TP + #FN (positives in the training data)
And #N = #FP+ #TN (negatives in the training data)

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 at its minimum, 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.

	$y_m = R(\vec{X}_m)$	$y_m \neq R(\vec{X}_m)$
$d(g(\vec{X}_m)+B \geq 0)$	True Positive (TP)	False Positive (FP)
$d(g(\vec{X}_m)+B < 0)$	True Negative (TN)	False Negative (FN)

2.4 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.

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

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.

2.5 F-Measure

The F-measures combine precision and recall into a single value. The F measures measure the effectiveness of retrieval. The best value is 1 when Precision and Recall are perfect. The worst value is at Zero.

F_1 Score:

$$F_1 = \frac{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}{2} = 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1 score is the harmonic mean of precision and recall.
The F_1 score weights recall higher than precision.

2.6 Accuracy

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

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

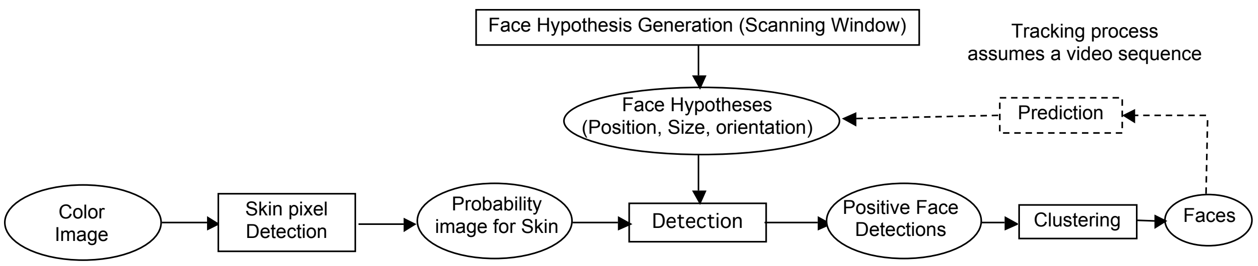
where M is the quantity of test data.

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.

3 Face Detection using Skin Color

Our first lab will use a detector for skin pixels to indicate possible faces. This process can be performed before the sliding window. 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.

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 grouped to form a single face detection.


Face Hypothesis Generation (Scanning Window)
Tracking process assumes a video sequence
Face Hypotheses (Position, Size, orientation)
Prediction
Color Image
Skin pixel Detection
Probability image for Skin
Detection
Positive Face Detections
Clustering
Faces

Algorithm:
1) Compute probability of skin at each pixel.
2) Test for faces at possible positions and sizes (scanning Window).
3) Cluster adjacent detections and estimate precise face position, size and orientation from clusters of detections.
Each step has can be done many different ways.

We can break the problem down into 3 challenges. You are asked to experimentally compare possible methods for each challenge:

Challenge 1: Detecting skin pixels with color
Challenge 2: Detecting Faces with Skin Color
Challenge 3: Face Localization

For each challenge we can propose a simple "baseline" method and possible better methods.

3.1 Challenge 1: Detecting skin pixels with color

Base Line: Ratio of RGB histograms

The first challenge is to transform a color (RGB) image, *X(i,j)* into an image where each pixel provides an estimate of the probability of skin, *P(i,j)*.

Assume a color image : $X(i,j) = \begin{pmatrix} R \\ G \\ B \end{pmatrix}(i,j)$

The algorithm will use a lookup table to convert color to probability.

$$P(i,j) \leftarrow L(\vec{X}(i,j))$$

The lookup table is constructed as a ratio of histograms, as explained below.
We can improve the results can be obtained by using a normalized color space as well as by tuning the quantization of the histogram to the data.

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.

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

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 $\underset{m}{\forall} h(\vec{X}_m) = h(\vec{X}_m) + 1$

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

For a color vector, $\vec{X} = \begin{pmatrix} R \\ G \\ B \end{pmatrix}$ we have two probabilities:

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

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 $L_{Skin}(\vec{X}) = \frac{h_T(\vec{X})}{h(\vec{X})}$

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 may provide better recognition.

In this example, $h(\vec{X})$ and $h_T(\vec{X})$ are composed of $2^8 \cdot 2^8 \cdot 2^8 = 2^{24}$ cells.
We define this as the Capacity of the histogram, V

$V = 2^8 \cdot 2^8 \cdot 2^8 = 2^{24}$ cells.

In general, $V = Q^D$ where Q is the number of values per feature and D is the number of features.

This can result in a very sparse histogram. The reliability can be improved by using more training images from more cases.

A naive statistics view says to have at least 10 training samples for histogram cell.
That is $M \geq 10\ V$. However, it is often easier to work with powers of 2.
For example, $2^3 = 8 \approx 10$ which is approximately 10.
This suggest that we required $M \geq 8\ V = 2^3\ V$.

Thus we would need $2^3 \cdot 2^{24} = 2^{27}$ training pixels. 2^7 Meg.
(Note that a 1024 x 1024 image contains 2^{20} pixels. This is the definition of 1 Meg)

Obtaining 2^7 Meg pixels is not a problem for $P(\vec{X}) = \frac{1}{M} h(\vec{X})$ but may be a problem for training the histogram of target pixels $P(\vec{X} \mid Skin) = \frac{1}{M_T} h_T(\vec{X})$.

A more realistic view is that the training data must contain a variety of training samples that reflect that variations in the real world.

What can we do? Two approaches:
We can reduce the number of values, Q, for each feature, or
we can reduce the number of features.

For example, for many color images, Q=32 color values are sufficient to detect objects. We simply divide each color R, G, B by 8.
R' = Trunc(R/8), G'=Trunc(G/8), B'=Trunc(B/8).

We can also use physics to look for features that are "invariant".

Variation: Detection with Chrominance

Luminance captures local surface orientation (3D shape) while Chrominance is a signature for object pigment (identity). The color of pigment for any individual is generally constant. Luminance can change with pigment density (eg. Lips), and skin surface orientation, but chrominance will remain invariant.

Several methods exist to transform the (RGB) color pixels into a color space that separates Luminance from Chrominance.

$$\begin{pmatrix} L \\ c_1 \\ c_2 \end{pmatrix} \Leftarrow \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

A popular space for skin detection is computed by dividing R and G by luminance. These are often called "r" and "g" in the literature.

Luminance: L= R+G+B

Chrominance : $r = c_1 = \frac{R}{R+G+B}$ $g = c_2 = \frac{G}{R+G+B}$

The terms r and g have values between 0 and 1. To count statistics we need integer values. For this, it is common to represent r and g as integer numbers coded with Q values between 0 and Q–1 by :

$$r = trunc\left((Q-1)\cdot\frac{R}{R+G+B}\right) \qquad g = trunc\left((Q-1)\cdot\frac{G}{R+G+B}\right)$$

From experience, Q = 32 color values seems to work well for skin with most web cameras, but this may change for certain data sets and should be experimentally verified.

Thus we can use a normalized vector $\vec{X} = \begin{pmatrix} r \\ g \end{pmatrix}$ as an invariant color signature for detecting skin in images.

Suppose we have a set of N training images $\{X_n(i,j)\}$ of size *RxC* where each pixel is an RGB color vector. This gives a total of $M=NxIxJ$ color pixels.

Suppose that M_T of these are labeled as skin pixels i.e. $y(X_n(i,j)) = P$
We allocate two tables: $h(r,g)$ and $h_T(r,g)$ of size Q xQ.
As before:

```
For all $i,j,n$ in the training set $\{X_n(i,j)\}$ :
BEGIN
    $r = trunc\left((Q-1)\cdot\frac{R}{R+G+B}\right)$ $g = trunc\left((Q-1)\cdot\frac{G}{R+G+B}\right)$
    $h(r,g) = h(r,g)+1$
    IF $(y(X_n(i,j) == P)$ THEN $h_T(r,g) = h_T(r,g)+1$ ; $M_T \leftarrow M_T + 1$
END
```

As before, we can obtain a lookup table $L_{skin}(r,g)$ that gives the probability that a pixel is skin. $L_{skin}(r,g) = \frac{h_T(r,g)}{h(r,g)}$

Given a new RGB image $C(i,j)$: For all i, j :

$$r = trunc\left((Q-1)\cdot\frac{R}{R+G+B}\right) \qquad g = trunc\left((Q-1)\cdot\frac{G}{R+G+B}\right)$$

$$P(i,j) = L_{skin}(r,g)$$

Where $P(i,j)$ is an image in which each pixel is a probability value from 0 to 1. Probabilities can be expressed, of course be expressed as integers by multiplying by a quantization value (Q).

Free Parameters

A number of parameters remain unspecified. Specification of these parameters generally depends on the application domain and the training data. Often these parameters make it possible to trade off error rates for computing time. To properly evaluate the algorithm, you should measure performance and compare performance over a range of parameters.

Free parameters for color based skin detection include:

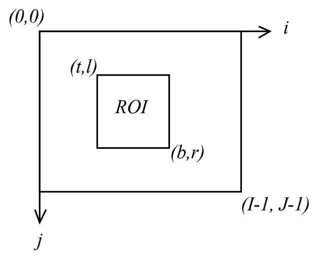
- The use of color coding (eg, RGB, rg, other codings for chrominance)

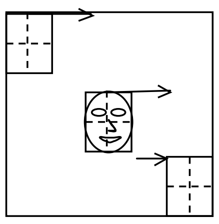
- Value of Q – the quantification for color color values.
- Training Data – different training and test regimes with variations such as the number of folds, and whether to train with a subset of the folds or all folds.

3.2 Challenge 2: Detecting Faces with Skin Color

We can detect faces from the mass of skin detections within a "Region of Interests" (ROI). The ROI is determined either by a-prior knowledge or by some form of search procedure. In many cases this is based on tracking of targets in the scene. This is not possible for our example, because our training and test data are restricted to static images.

In many vision detectors the ROI is simply a sliding window that scans the entire image as explained above. This is the technique used, for example, with the Viola Jones Face Detector.





A common representation for the ROI is a rectangle represented by four coordinates: (left, top, right, bottom) or (*l, t, r, b*). Alternatively the center, width and height of the ROI: c_i, c_j, w, h.

For detection in static images, we will need to test a range of ROIs with different positions and sizes. The scanning step size (s) may also be varied. This may be the same size for row and column direction, or different step sizes may be used (say s_i and s_j)

Baseline: Sliding Window Detector

We can compute the ROI parameters for a face as a bounding box for an ellipse.

Let us define a face hypothesis as $\vec{X} = \begin{pmatrix} c_i \\ c_j \\ w \\ h \end{pmatrix}$

Since faces are generally vertical, we may assume that the ellipse is oriented with the major axis aligned with the row direction (j). The bounding box ROI is:

$$t = c_j - \frac{h-1}{2}, \qquad b = c_j + \frac{h-1}{2}, \qquad l = c_i - \frac{w-1}{2}, \qquad r = c_i + \frac{w-1}{2}$$

The likelihood of a face at a position *(c_i, c_j)* of size *(w,h)* is:

$$g(\vec{X}) = \frac{1}{w \cdot h} \sum_{i=l}^{r} \sum_{j=t}^{b} P(i,j)$$

We can bias this likelihood to make a decision:

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

And of course

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

This technique will detect "faces" for a range of positions and sizes. As long at the detections overlap with the face ellipse in the data-base they are TRUE detections.

The problem with this technique is that faces are not square. The ROI gives equal weight to corners as the center. We can do better by weighting the pixels using a Gaussian function. This is called a "robust estimator" because it tends to reject outliers.

Variation: Detection using a Gaussian mask.

In machine vision, fixation serves to reduce computational load and reduce errors by focusing processing on parts of the image that are most likely to contain information. A commonly practice is to use a Gaussian Window to suppress signals outside of the fixated region. A similar technique is used to suppress outliers for robust estimation.

A Gaussian window has the form:

$$G(\vec{P};\vec{\mu},\Sigma) = e^{-\frac{1}{2}(\vec{P}-\vec{\mu})^T \Sigma^{-1} (\vec{P}-\vec{\mu})}$$

Where $\vec{P} = \begin{pmatrix} i \\ j \end{pmatrix}$ represents image positions, and $\vec{\mu} = \begin{pmatrix} \mu_i \\ \mu_j \end{pmatrix}$ is the center position of the window.

The covariance matrix of the window is: $\Sigma = \begin{pmatrix} \sigma_i^2 & \sigma_{ij} \\ \sigma_{ij} & \sigma_j^2 \end{pmatrix}$

The coefficients have a value of 1 at the center of the mask, and taper to 0.1 at a distance of 2σ and to 0.01 at a distance of 3σ.

We can use the parameters of the window to define a face hypothesis. $\vec{X} = \begin{pmatrix} \mu_i \\ \mu_j \\ \sigma_i^2 \\ \sigma_j^2 \\ \sigma_{ij} \end{pmatrix}$

Normally the Gaussian mask should be computed within a ROI determined by a bounding box. Typically the bounding box should be at least 2σ (standard deviations), but if speed is more important the false negatives, then computation time can be reduced by using a smaller ROI, down to as small as 1 standard deviation.

We can define the ROI as a 2σ bounding box:

$$t = c_j - 2\sigma_j, \qquad b = c_j + 2\sigma_j, \qquad l = c_i - 2\sigma_i, \qquad r = c_i + 2\sigma_i$$

To evaluate a face hypothesis, we compute the face likelihood as a weighed sum of the skin probabilities by the Gaussian mask.

$$g(\vec{X}) = \sum_{i=l}^{r} \sum_{j=t}^{b} P(i,j) G(i,j;\vec{X})$$

As before, the discriminant, $g(\vec{X})$, has a value between 0 and 1. This is the likelihood that a face may be found at $\vec{X}$.

As before, faces are detected as:

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

Free parameters to test

As with color skin detection a number of parameters remain unspecified. To properly evaluate the algorithm, you should measure performance and compare performance over a range of parameters.

Free parameters for color face detection include:
For a simple scanning window, these include:

- Width and height of ROI
- Range of positions
- Step size for scanning windows
- The percentage of overlap with the ground truth that is considered a TRUE detection.

For a Gaussian detection window, parameters also include

- The width and height of the ROI compared to the standard deviations of the principal axis.
- The range and step sizes for orientation of the Gaussian window.

3.3 Challenge 3: Face Localization

Detection vs Localization

<u>Detection</u>: A face is considered to be "detected" if a positive detection is found at a position that overlaps the face in the ground truth. The simplest form of test is to verify that the location of the face is within the ellipse of the ground truth label for a face in the image.

For a face hypothesis at $\vec{X} = \begin{pmatrix} c_i \\ c_j \\ w \\ h \end{pmatrix}$ a face is detected if $R(\vec{X})$=P

The detection is TRUE if $R(\vec{X})=y(\vec{X})$

Using the ground truth $y(\vec{X}) = \begin{cases} P & if(\frac{(x-c_x)^2}{r_a^2} + \frac{(y-c_y)^2}{r_b^2} \leq 1 \\ N & otherwise \end{cases}$

The Hypothesis is a TRUE POSITIVE detection if $y(\vec{X})$=P and $R(\vec{X})=y(\vec{X})$.

A large number of TP faces will be detected for each ground truth face. If the search space includes multiple scales and orientations, than the number of detected faces will be even larger. All of these correspond to the same face!

Localization (more precisely parameter estimation) specifies precisely where, and with what parameters, the face may be found. There should be only ONE face located for each true face. The face should be located at a position, size and orientation as close to the true face as possible. When searching at multiple scales and orientations, each detection has the form of an error vector.

A distance metric, relating degrees and scale change to position is required to reduce this to a single number. For discrete samples the distance metric can provided implicitly by the sample step size in scale, orientation and position.

We can obtain a location (or parameter estimation) from multiple positive detections by suppressing detections for which the discriminant, $g(\vec{X})$ is not a local maximum. This requires specifying a measure for locality.

We can also estimate the parameters of the face from the moments of a "cloud" of detections at multiple positions, scales and orientations. This is the same robust estimation technique that we used above, extended to robustly estimate position, size and orientation.

Baseline: Localization by non-maximum suppression

In order to suppress non-maximal detections, the easiest method is to build a list of detections hypotheses, $\{\vec{X}\}$ over the desired range of positions, scales, orientations and any other parameters, and then filter this list to remove any hypothesis for which the discriminant not a local maximum. (A maximum within a some distance R.)

$\forall \vec{X}$
$\forall \vec{X}_i, \vec{X}_j \in \{\vec{X}\}$: IF $Dist(\vec{X}_i, \vec{X}_j) < R$ AND $g(\vec{X}_i) < g(\vec{X}_j)$ THEN $\{\vec{X}\} \leftarrow \{\vec{X}\} - \vec{X}_i$

This requires defining some notion of distance that includes position, scale and orientation. This is generally done with regard to an expected range of positions, orientations and scales over which a single face would be detected. For example, using a Mahalanobis distance (Distance normalized by covariance).

Variation: Localization by Robust Estimation

We can use robust estimation to estimate the most likely parameters from a set of detections. To do this, we will calculate the weighted moments from the set of detections.

Suppose that we have a set of N detections: $\{\vec{X}_n\}$ and that for each detection we have a discriminant $g(\vec{X}_n)$.

The mass of the detections is $M = \sum_{n=1}^{N} g(\vec{X}_n)$. This is the zeroth moment of the set of detections.

The "expected value" is $E\{\vec{X}\} = \frac{1}{M}\sum_{n=1}^{N} g(\vec{X}_n)\cdot \vec{X}_n$

This is the first moment (or center of gravity) of the values of $\{\vec{X}_n\}$.

The expected value is a vector of first moments for each parameter:

For D parameters, the center of gravity is a vector

$$\vec{\mu} = E\{\vec{X}\} = \frac{1}{M}\sum_{n=1}^{N} g(\vec{X}_n)\vec{X}_n = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \dots \\ \mu_D \end{pmatrix} = \begin{pmatrix} \frac{1}{M}\sum_{n=1}^{N} g(\vec{X}_n)X_{1n} \\ \frac{1}{M}\sum_{n=1}^{N} g(\vec{X}_n)X_{2n} \\ \dots \\ \frac{1}{M}\sum_{n=1}^{N} g(\vec{X}_n)X_{Dn} \end{pmatrix}$$

The vector $\vec{\mu}$ the vector of weighted averages for the components of $\vec{X}_n$.

If there is only one face inside the set $\{\vec{X}_n\}$ of positive detections then $\vec{\mu}$ provides the most likely estimate for set of parameters the detection, (e.g. position, size and orientation).

In the case of multiple faces, the best estimate for each of the faces can be determined using the Expectation Maximization (EM) algorithm. Alternatively, the estimation may be limited to faces detections within a small region of the image.

Free parameters to test

As with color face detection a number of parameters remain unspecified. To properly evaluate the algorithm, you should measure performance and compare performance over a range of parameters.

Free parameters for color face detection include:
For a simple scanning window, these include:

- Number of Faces in the image
- Range of positions, size and orientations to test
- Distance to use for non-maximal suppression.
- Size of the region used to cluster faces detections.