Computer Vision

James L. Crowley and Nachwa Aboubakr

MoSIG M2 Fall Semester 2020
Lesson 1 1 October 2020

Performance Evaluation for Pattern Detection and 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. Course Organisation

The MoSIG M2 Course on Computer Vision has two parts. Part 1: Image Analysis and Understanding, taught by James Crowley, and Part 2 on 3D Vision and Motion analysis taught by Edmond Boyer. This is the introduction for the first half of the class.

The first six lectures on Image Analysis and Understanding will be evaluated with a report based on a running series of programming exercise. Each lesson will be accompanied by a programming exercise, programmed in Python using OpenCV and Keras.

The common thread for these exercises is comparative performance evaluation. We will use the problem of face detection as a running example to compare different techniques for detecting, classifying and tracking information in images. For each lecture, you will be asked to implement a form of face detector, and evaluate its performance using a standard publicly available data set.

Exercises will be performed in teams of 2 and handed in as a jupyter Notebook within 1 week of the lecture. Exercises will not be graded, each time will receive written feedback. Each practical exercise will build on the exercise of the pevious week. Thus it is important to do the practical exercises each week and not wait until the midterm. The midterm grade (50%) will be based on a written report providing the results of comparative performance evaluation for the techniques covered in the programming exercises.

Each lecture will be composed of 1h30 to 2h00 of computer vision theory, followed by 1h00 to 1h30 of practical instruction using Open CV.

Oct 1 Lesson 1: Pattern Detection and Performance Evaluation
Practical 1: Use Python and OpenCV to display faces from the FDDB data set.

Oct 15 Lesson 2: Visual Perception in Man and Machine
Practical 2: Face Detection with a sliding window multi-layer perceptron

Oct 22 Lesson 3: Scale Space and Image Pyramids
Practical 3: Face detection at multiple scales with a multi-layer perceptron

Nov 5 Lesson 4: Gaussian Receptive Fields, HOG and SIFT

Practical 4: Face detection at multiple scales with a CNN

Nov 12 Lesson 5: Robust Bayesian Tracking and the Kalman Filter
Practical 5 Real Time Face Detection and Tracking

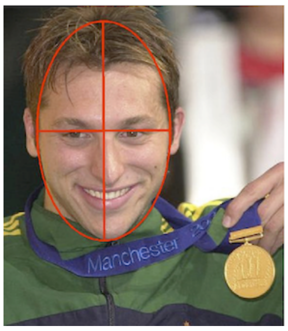
Nov 19 Lesson 6: Attention and Cognition for Computer Vision

Final project: Performance evaluation for Face Detection and Tracking

Tools for programming exercises
Lab exercises will be implemented and reported by teams of 2 students submitted each week at Jupyter Notebooks. Team compositions are to be determined should be finalized by the start of lecture 2 on the 14th of Oct. Each time will submit a written report about the programming exercises, describing the results of comparative performance evaluation for the different techniques. The midterm grade, 50% of the final grade, will be based on the written report. This report should evaluate the effects of changing parameters for each technique and given an overall comparison of techniques. Groups will be encouraged to be creative in implementing, evaluating and reporting the labs.

Programming exercises will be performed using the OpenCV environment running under Python. This morning, Nachwa Aboubakr assisted by Yangtao Wang will discuss how to install Open CV running under Python, and how to load and display an image from the FDDB benchmark data set of images.

Several sets of annotated images of faces are available on the course web site.
IN general we will use the FDDB (Face Detection Data Set and Benchmark) data set maintained at UMASS: http://vis-www.cs.umass.edu/fddb/, but other data sets are allowed. In FDDB, images are RGB with each pixel containing 3 colors: Red, Green and Blue. Face regions have been hand-labeled with ellipses to indicate the position of faces. A typical image in FDDB with its annotated face region looks like this.



Note that the major axis of the ellipse is 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 $y(\vec{X}_m)$

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

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

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

2. Pattern Recognition and Machine Learning

Pattern Recognition 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".

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

A classic example of a sensor is a camera, but many forms of sensors are possible. Features may be Boolean, natural numbers, integers, real numbers or symbolic labels. In most interesting problems, the sensor provides a vector of D features, $\vec{X}$.

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

A classifier, $R(\vec{X})$, maps the feature vector, $\vec{X}$ 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.

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: $\vec{X} \to R^K$

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.

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

A pattern detector is a form of classifier that returns a yes/no decision. This is a special case of recognition.

For a pattern detector, 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.

Machine Learning for Pattern Recognition

Machine learning explores the study and construction of algorithms that can learn functions from data. Learning for Pattern Recognition is the most common form of machine learning, but this is only one of many forms of machine learning. Over the last 50 years, machine learning techniques have been developed for many different problems involving function estimation. For example, machine learning techniques have been demonstrated from generating images, videos, speech, and music as well as for controlling robots (manipulation, walking, expressing emotions).

For pattern recognition, the common approach is to use a set of "training data", $\{\vec{X}_m\}$, to estimate the discriminant function $\vec{g}(\vec{X})$. A variety of algorithms have been developed, each with its own advantages and disadvantages.

Most classical methods use a discriminant function. This is typically learned 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 quality of the recognizer depends on the degree to which the training data $\{\vec{X}_m\}$ represents the range of variations of real 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 techniques learn the recognition function without a labeled training set. Such methods typically require a much larger sample of data for learning. A number of hybrid algorithms exist that initiate learning from a labeled training set and then extend the learning with unlabeled data.

3. Performance Evaluation for Pattern Recognition

Machine learning is an empirical science. New techniques are continuously introduced with rapid progress in reliability. To publish a technique it is necessary to demonstrate a gain in performance compared to previous techniques, but demonstrating performance evaluation metrics on a publicly available benchmark data set.

As mentioned above, 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 discriminant 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 discriminant 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 combine the discriminant with a decision function to define a classifier, $R(\vec{X}_m)$.

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

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

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 the bias term, B, is swept through a range of values, it changes the ratio of true positive detection to false positives.

If, for example, the discriminant functions $g_1(\vec{X})$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 $\{\vec{X}_m\}$, $\{y_m\}$.

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).

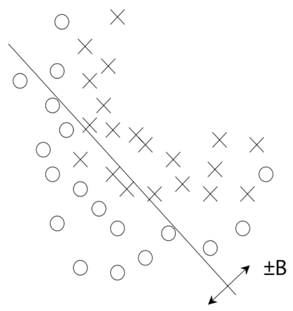
For the M samples of the training data $\{\vec{X}_m\}$, $\{y_m\}$ we can define:
#P as the number of Positives in the training data.
#N as the number of Negatives in the training data.
#T as the number of training samples correctly labeled by the detector.
#F as the number of training samples incorrectly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

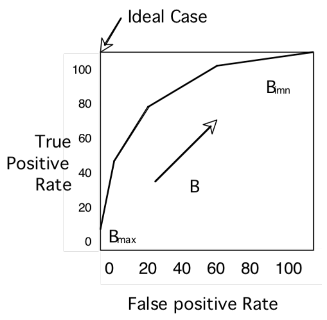
Note that #P = #TP + #FN (positives in the training data)
And #N = #FP+ #TN (negatives in the training data)

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)

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.

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:

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

The F1 score is the harmonic mean of precision and recall.

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.