Pattern Recognition and Machine Learning

James L. Crowley and Nachwa Aboubakr

ENSIMAG 3 - MMIS — Fall Semester 2020
Lesson 1 — 7 October 2020

Performance Evaluation

Outline

Notation

x_d	A feature. An observed or measured value.
$\vec{X}$	A vector of D features.
D	The number of dimensions for the vector $\vec{X}$
K	Number of classes
C_k	The kth class
$\vec{X} \in C_k$	Statement that an observation $\vec{X}$ is a member of class C_k
$\hat{C}_k$	The estimated class
$R(\vec{X})$	A Recognition function
$\hat{C}_k = R(\vec{X})$	A recognition function that predicts $\hat{C}_k$ from $\vec{X}$
	For a detection function (K=2), $C_k \in \{P,N\}$
y	The true class for an observation $\vec{X}$
$\{\vec{X}_m\}$ $\{y_m\}$	Training samples of $\vec{X}$ for learning, along with ground truth $\vec{y}$
y_m	An annotation (or ground truth) for sample m
M	The number of training samples.

1. Course Organisation

This course gives an introduction to techniques for Pattern Recognition and Machine Learning, with a performance based approach, using detection of faces as running example. The course takes a performance based approach, requiring a series of programming projects performed in Python using OpenCV and Keras. Projects are performed in teams and focus on experimental performance evaluation of techniques using benchmark data sets. Access to a personal computer for use in projects is strongly advised. All lectures are in given in English. Labs my be reported in French or English.

In the introductory lecture we introduce the problem of pattern recognition, and review performance evaluation metrics for pattern detection. We also discuss the use of Conda Python, OpenCV and Jupyter Notebooks for practical exercises. In the following lectures we review sliding window pattern detectors and discuss the use of the Viola-Jones face detector found in OpenCV.

The course will then focus on the design and training of Artificial Neural Networks. We introduce the Perceptron and discuss training with gradient descent. We present multi-layer networks and derive the Back-propagation algorithm as a distributed algorithm for gradient descent. We present convolutional neural networks, generative networks and Support Vector Machine.

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 the Viola Jones cascade detector
Lab 2: Face detection using multilayer fully connected neural networks.
Lab 3: Face detection using convolutional 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. With 30 students we will have 3 presentations for 2 of the labs,

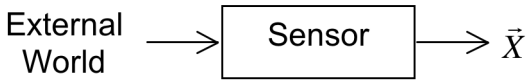
Labs will be performed using the OpenCV environment running under conda Python. Groups will be encouraged to be creative in implementing, evaluating and reporting the labs. The labs may use code found downloaded from the internet PROVIDED that you document the origin of the code. The primary task is 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. 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 transforms some physical phenomena into one or more measurements, $\vec{X}$.



These measurements are classically called features and $\vec{X}$ is called a feature vector. Features may be Boolean, natural numbers, integers, real numbers or symbolic labels.

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

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

Discriminant and Decision Functions

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

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

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

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

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

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

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

The discriminant function is typically learned from the data.

The classifier "guesses" or "predicts" the most likely class $\hat{C}_k \leftarrow R(\vec{X}) = d\left(\vec{g}\left(\vec{X}\right)\right)$

The discriminant function 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.

Machine Learning for Pattern Recognition

Machine learning explores the study and construction of algorithms that can learn functions from data. Machine Learning for Pattern Recognition is the most common form of Machine Learning, but this is only one of many forms. Over the last 50 years, machine-learning techniques have been developed for many different problems involving function estimation, including speech synthesis, music and art.

Machine learning uses a set of set of M samples $\{\vec{X}_m\}$, to estimate the discriminant function $\vec{g}\left(\vec{X}\right)$. A variety of algorithms have been developed, each with its own advantages and disadvantages.

Classic techniques for machine learning use probability theory to make this prediction.

$$\hat{C}_k = \arg\!-\!\max_{C_k}\left\{P(C_k \mid \vec{X})\right\}$$

where $P(C_k \mid \vec{X})$ is the conditional probability of the class C_k given the vector $\vec{X}$.

In this case the decision function $d(-)$ is $\arg\!-\!\max\{-\}$ and the discriminant function, $\vec{g}\left(\vec{X}\right)$ is the conditional probability $P(C_k \mid \vec{X})$.

We can use Bayes Rule to estimate $P(C_k \mid \vec{X})$: $\quad P(C_k \mid \vec{X}) = \dfrac{P(\vec{X} \mid C_k)P(C_k)}{P(\vec{X})}$

Our problem is then reduced to using the training data to estimate the probabilities: $P(C_k), P(\vec{X})$, and $P(\vec{X} \mid C_k)$.

This can be done with a variety of parametric and non-parametric techniques. The most widely used parametric models include the multivariate Gaussian (normal) density and the Gaussian Mixture model:

Gaussian (Normal) Density: $p(\vec{X}) = \mathcal{N}(\vec{X}; \vec{\mu}, \Sigma) = \dfrac{1}{(2\pi)^{\frac{D}{2}} \det(\Sigma)^{\frac{1}{2}}} e^{-\frac{1}{2}(\vec{X}-\vec{\mu})^T \Sigma^{-1}(\vec{X}-\vec{\mu})}$

Gaussian Mixture Models: $p(\vec{X}) = \sum_{k=1}^{K} \alpha_k \mathcal{N}(\vec{X}; \vec{\mu}_k, \Sigma_k)$

Non-parametric techniques use the training data as a model for the probability. These include Histograms, Kernel Density Estimators, and K-Nearest Neighbors.

I assume that you have seen such techniques in your first two years at ENSIMAG. We will concentrate on more modern technique based on Neural Networks.

Supervised Learning: Having the true class $\{y_m\}$ for each of the *M* training samples, $\{\vec{X}_m\}$, makes it much easier to estimate the functions $g_k(\vec{X})$**.** This is known as supervised learning.

Unsupervised Learning techniques learn the discriminant function $g_k(\vec{X})$ 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.

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, using a publicly available benchmark data set.

3. Performance Evaluation

We will illustrate performance evaluation with pattern detection. 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). The possible classes are $C_k \in \{P,N\}$

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

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

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

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

Observations for which $g(\vec{X}) \le 0$ are estimated to be members of the background. This will be called NEGATIVE or N. (N includes the Neutral class where $g(\vec{X}) = 0$.)

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

$$R(\vec{X}) = d(g(\vec{X})) = \begin{cases} P & \text{if } g(\vec{X}) > 0 \\ N & \text{if } g(\vec{X}) \le 0 \end{cases}$$

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
$R(\vec{X}_m) = y_m$ AND $R(\vec{X}_m) = N$ is a TRUE NEGATIVE or TN
$R(\vec{X}_m) \neq y_m$ 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

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 as the bias term, B, is swept through a range of values.

For example, if $g(\vec{X}_m)$ is a probability ranging from 0 to 1, the decision function would be

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

in this case, B can range from -0.5 to more than $+0.5$.
When $B = -0.5$ 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 ROC plots True Positive Rate (TPR) against False Positive Rate (FPR) as a function of B for the training data $\{\vec{X}_m\}$, $\{y_m\}$.

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.

In some practical cases, we can replace B with any adjustable parameter for the discriminant function. Thus we can use the ROC to determine the optimum parameter values for an architecture for machine learning. We can also use an ROC curve to compare architectures.

True Positives and False Positives

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

IF $g(\vec{X}_m)+B > 0.5$ 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 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:

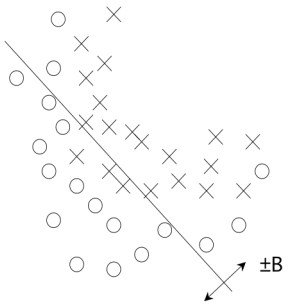
#TP as the number of training samples correctly labeled as Positive
#FP as the number of training samples incorrectly labeled as Positive
#TN as the number of training samples correctly labeled as Negative
#FN as the number of training samples incorrectly labeled as Negative

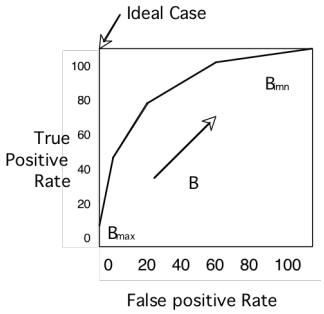
Note that #P = #TP + #FN (positives in the training data)
And #N = #FP+ #TN (negatives in the training data)

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.

	$y_m = R(\vec{X}_m)$	$y_m \neq R(\vec{X}_m)$
$d(g(\vec{X}_m)+B > 0.5)$	True Positive (TP)	False Positive (FP)
$d(g(\vec{X}_m)+B \leq 0.5)$	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.

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

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.

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.

4. Tools and Data Sets

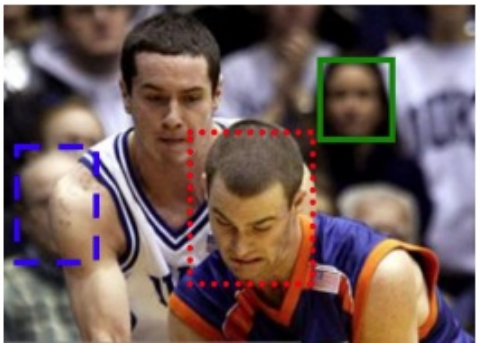
Programming exercises should be performed using the OpenCV environment running under Python. You should use Jupyter notebooks to perform your project. However, your project reports should be handed in as written report written in English or French, and describing performance evaluation under different configurations and parameters.

To install Open CV running under Python, go to the conda web site at (https://docs.conda.io/en/latest/miniconda.html), download the shell script for miniconda and follow the installation instructions. Create a workspace (for example named Machine-Learning), and install Juypiter notebooks and OpenCV. The following is a trace of these steps from an Mac OS. The actual commands will depend on your operating system.

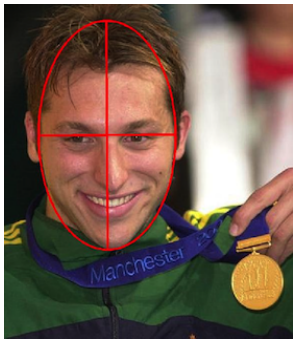
```
> bash /Users/Crowley/Downloads/Miniconda3-latest-MacOSX-x86_64.sh
> conda
Open a new terminal window
> conda create -n Machine-Learning
> conda activate Machine-Learning
> conda env list
> conda install -c conda-forge opencv
> conda install -c conda-forge jupyter
> conda install matplotlib
> python
Python 3.8.5 | packaged by conda-forge | (default, Sep 16 2020, 17:43:11)
[Clang 10.0.1 ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> import numpy
>>> jupyter notebook
```

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 Home” of the University of Massachusetts. The data set can be found at http://vis-www.cs.umass.edu/fddb/ and is described in the paper (Jain and Learned-Miller 2010) available on the course web site.
http://crowley-coutaz.fr/jlc/Courses/2020/GVR.VO/GVR-VO.html

The FDDB data set contains 2845 images with a total of 5171 faces extracted from news articles selected using an automatic face detector. Faces with a height or width less than 20 pixels, as well as faces that are looking away from the camera were rejected. The remaining 5171 faces have been noted in a ground-truth data set and labeled with a bounding box. The images in this data set exhibit large variations in pose, lighting, background and appearance due to factors such as motion, occlusions, and facial expressions, which are characteristic of the unconstrained setting for image acquisition. Each face is also described with an ellipse parameterized by center location, the lengths of its major and minor axes, and its orientation, as shown in the following images:



Face Boxes



Face Ellipses

Annotations of face regions as an ellipse in FDDB are represented 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. For example:

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 $y(\vec{X}_m)$

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

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

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