Pattern Recognition and Machine Learning

James L. Crowley

ENSIMAG 3 MMIS First Semester 2010/2011

Lesson 1 29 September 2010

# Course Overview and Introduction

**Contents**

Source:
"Pattern Recognition and Machine Learning", C. M. Bishop, Springer Verlag, 2006.

## Notation

| | |
|---|---|
| x | a variable |
| X | a random variable (unpredictable value) |
| $\vec{x}$ | A vector of D variables. |
| $\vec{X}$ | A vector of D random variables. |
| D | The number of dimensions for the vector $\vec{x}$ or $\vec{X}$ |
| E | An observation. An event. |
| k | Class index |
| K | Total number of classes |
| $\omega_k$ | The statement (assertion) that E $\in T_k$ |
| $M_k$ | Number of examples for the class k. (think M = Mass) |
| M | Total number of examples. $M = \sum_{k=1}^{K} M_k$ |
| $\{X_m^k\}$ | A set of $M_k$ examples for the class k. $\{X_m\} = \bigcup_{k=1,K} \{X_m^k\}$ |
| $\{t_m\}$ | A set of class labels (indicators) for the samples |

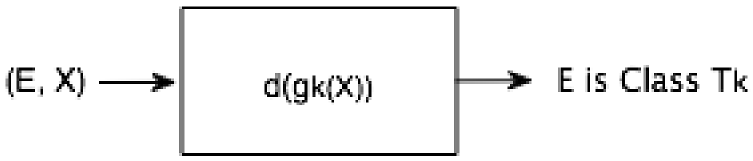
## The Pattern Recognition Problem

Le E be an observation event that is described by some number of "features", $\vec{X}$

$(E, \vec{X})$

Let us define K classes of events $\{T_k\}$. for k=1 to K.
Let $\omega_k$ represent the proposition that the event E $\in$ the class k

Our problem is to build a box that maps a set of features $\vec{X}$ into a class $T_k$ from a set of K possible Classes.



Let $\omega_k$ be the proposition that the event belongs to class k: $\omega_k = E \in T_k$

The classification function can be decomposed into two parts: d() and $g_k()$:

$$\hat{\omega}_k = d(y(\vec{X}))$$

where :

$$\vec{g}(\vec{X}) = \begin{pmatrix} g_1(\vec{X}) \\ g_2(\vec{X}) \\ \dots \\ g_K(\vec{X}) \end{pmatrix}$$ A set of discriminant functions : $R^D \rightarrow R^K$

and d() : a decision function $R^K \rightarrow \{\omega_K\}$

This class is essentially about techniques to learn g(X) and d(g(X)).

## Course Overview

Lesson 1: Introductory Concepts
- Introductory Example
- Curve Fitting

Lesson 2: Bayesian Probability Theory
- Probability
- Bayesian Probability
- Probability Density Functions

Lesson 3: Linear Models for Regression
- Decision Theory
- Information Theory
- Linear Bases and Least Squares
- Bias-Variance decomposition
- Bayesian Linear Regression
- Evidence Approximation

Lesson 4 Linear Models for Classification
- Discriminant Functions
- Least Squares Discrimination
- Fisher Discrimination
- The Perceptron Algorithm

Lesson 5 Probabilistic Generative Models
- Maximum Likelihood
- Discrete Features

Lesson 6 Probabilistic Discriminative Models
- Fixed Bases
- Logistics Regression
- Iterative Least Squares
- Multiclass logistics
- Probit regression

Lesson 7 Bayesian Logistic Regression
- Laplace Approximation
- Predictive Distribution

Lesson 8 Kernel Methods

- Dual Representations
- Constructing Kernels
- Radial Basis Functions

Lesson 9 Gaussian Processes

- Linear regression (again)
- Gaussian processes for regression
- Learning Hyper-Parameters
- Relevance Determination
- Gaussian Processes for Classification

Lesson 10 Support Vector Machines

- Relation to Logistics Regression
- Multiclass SVMs

Lesson 11 Relevance Vector Machines

- RVM for regression
- RVM for classification

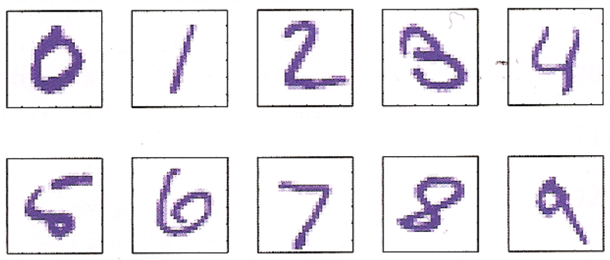
Lesson 12 Principle Components Analysis

- Maximum Variance Formation
- Minimum error Formulation
- Applications

# Introduction

## An Example: Recognizing Handwritten Digits

Consider a 28 x 28 pixel imagette, *X(i,j)* containing handwritten digits,
tel que $i \in [1, I], j \in [1, J]$,



Each imagette, X(i,j) can seen as a vector of 748 real numbers, *X(d) =X(i,j)*
where $d = j{\cdot}I + i$.

Our goal is to map an observation of *X(i,j)* to one of 10 classes representing the 10 digits $T_k = \{0, 1, 2, 3, \ldots, 9\}$.

We could use heuristics or structural models to build an algorithm to recognize each digit. Experience shows that the great variety of handwriting styles leads to a multitude of special cases and poor, unreliable results. 50 years of research has failed to produce such a method. Such hand crafted algorithms are always unreliable when faced with real handwriting.

However, over the last 20 years, Bayesian probabilistic methods have provided a variety of elegant and reliable solutions.

***Image Analysis vs Recognition***:
Detecting, extracting and delimiting and normalizing the size of each digit is a problem of image analysis. For digits, we CAN build algorithms that work reasonably well. Such algorithms are the examined in the course Image Formation and Analysis.

Even for image analysis, in many cases the most reliable analysis is simply to attempt to recognize a pattern at every possible position, and orientation (brute force).
This is called a scanning window approach, and can work when there are a small number of classes to detect.

For example, this is used for face detection (we will see this in Formation and Analysis of Images). Reliable classification requires machine learning methods, such as those that we will examine in this class.

**Basic Concepts.**

***Training***

Suppose a training set of M imagettes: $\{\vec{X}_m\}$.
By inspection we can build a label for each imagette $\{t_m\} \in T_k = \{0, 1, 2,\ldots, 9\}$
The set $\{t_m\}$ is sometimes referred to as "ground truth".

We then use a <u>machine learning</u> algorithm to build a function: $y(\vec{X}_m)$.
This function is learned in a <u>training phase</u> using the M examples $\{\vec{X}_m\}$ and $\{t_m\}$.

***Generalisation***

We must then test the learned function with a test set $\{\vec{X}_m\}$, $\{t_m\}$.

The test set must be different from the training set.
The ability to learn a model y() that successfully classifies the test set is called "Generalisation". This is a central problem in Pattern Recognition.

***Supervised Learning***

Applications where the training data comprise both $\{\vec{X}_m\}$ and $\{t_m\}$ are called "Supervised learning".

***Regression vs Classification***

When the goal is to assign a vector to one of a finite (discrete) set of classes this is called "classification".

If the goal is to assign the vector to a continuous variable, this is called "regression". Regression is used for problems such a process control for determining the appropriate mixtures of chemicals, or the settings for reference parameters.

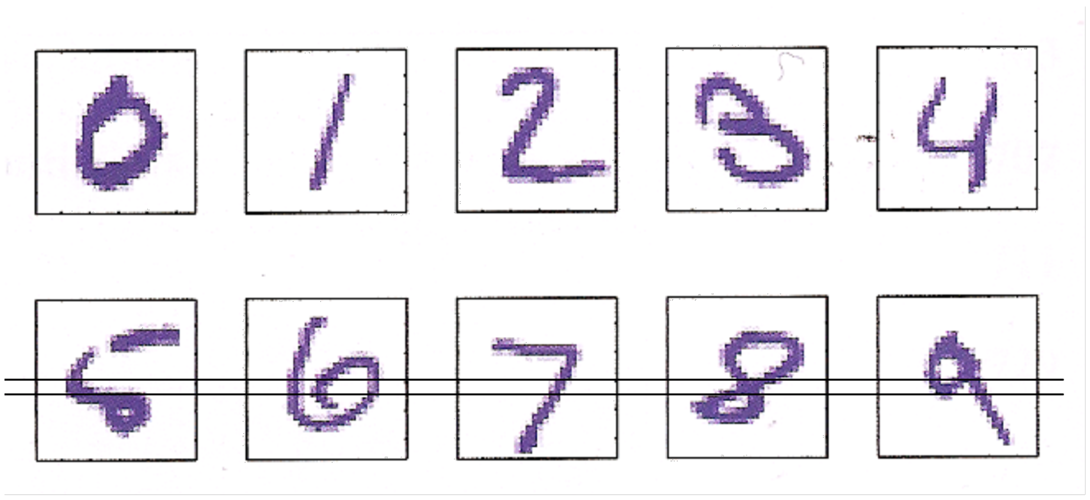
***Unsupervised Learning***

If learning is performed using only $\{\vec{X}_m\}$ without labels, then this is "Unsupervised Learning". Unsupervised learning is often used for clustering, for data analysis when the number of classes is not known in advance. Unsupervised learning could tell us how many digits are used in written text, and provide prototypes for each digit.

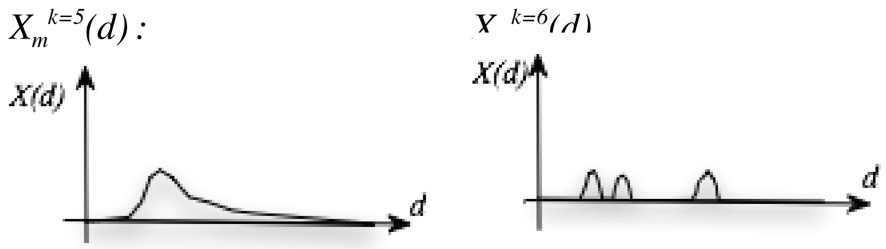
***Reinforcement Learning***
A third kind of learning is "Reinforcement learning". Reinforcement learning seeks to determine a suitable sequence of actions by trying out different actions and assigning reward or penalties according to success. We will not discuss Reinforcement Learning in this course (although it is very important in my own research).

**Curve Fitting**
Many of the concepts of machine learning can be illustrated using the problem of curve fitting.

For example, one approach to recognize our digits would be to fit a curve in a 768 dimensional space. For illustration, lets consider fitting a curve to as single row of our digits.



For example, for digits k= 5 and k=6 an example curve might be.



Consider a simple function on M observations of $\mathbf{X} = \{X_m\}$, where each observation is labeled with a value $\mathbf{t} = \{t_m\}$ for m = 1 to M where $t_m$ is from the class {0,1,2,...,9} Our goal is to learn a model y-k() that can map each $X_m$ onto $t_m$.

As a simple example, consider the 12th row of each imagette so that *X(d) = X(i,12)* for *d = 1 to 28*

Suppose that we have $M_k$ examples for each class, *k*, of digit: $\{ X_m^k \}$
and that we have a set of $\{t_m\}$ labels for each example.

We wish to learn a "generative" model, $y_k(d)$ for the curves for each digit, k.

One such model would be an N^th order polynomial

$$y(d,\vec{W}) = w_o + w_1\, d + w_2\, d^2 + \ldots + w_N d^N$$

for *d = 1 to 28*. D=28.

$$y(d,\vec{W}) = \sum_{n=0}^{N} w_n \cdot d^n$$

where $w$ is a coefficient to be learned and $i^n$ represents the position (integer) raised to the power *n*. Let $\vec{W}$ represent the vector of weights

$$\vec{W} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_N \end{pmatrix}$$

For any example we can determine an "error" function, $E(\vec{X},\vec{W})$,

$$E(\vec{X},\vec{W}) = \frac{1}{2}\sum_{d=1}^{D} \{y(d,\vec{W}^*) - X\ (d)\}^2$$

For our training set, the error is the sum of all errors.

$$E(\vec{W}) = \sum_{m=1}^{M_k} E(\vec{X}_m,\vec{W})$$

The result is a non-negative error function that is 0 when the curve fits all the data. Because this is a squared error function, we can use it to analytically solve for the best $\vec{W}$ by setting the derivatives to zero.

Let us represent the optimum (smallest error) vector as $\vec{W}^*$.

Thus a generative model for each class k is $y_k(d) = y(d, W_k^*)$

To determine the class we simply look for

$$\min_k \{ \frac{1}{2} \sum_{m=1}^{M_k} \sum_{d=1}^{D} \{y_k(d) - X(d)\}^2$$

in this example $y_k(d)$ is a discriminative function.

***RMS Error.***

Notice that the error $E(W)$ grows with the number of examples.

$$E(\vec{W}) = \frac{1}{2} \sum_{m=1}^{M_k} \sum_{d=1}^{D} \{y(d, \vec{W}^*) - X_m(d)\}^2$$

A better measure would normalize for the number of examples. This is provided by a "Root Mean Square" (RMS) error.

$$E_{RMS}(\vec{W}) = \sqrt{2E(\vec{W}) / M_k}$$

**The Model Selection Problem**

A classic problem in learning is to select the model.  In or example this was reduced to selecting the order of the polynomial, N.

For example, for any example, X(d) we can learn a perfect fit by setting N=D.
We could even learn a function for M samples by setting $N=M_kD$

This would not generalize well to other examples!.

In this case we would be better served to use the data as a model, using, for example, Parzen windows or Gaussian windows.

In general, model selection includes

1) Selection of the family of basis functions for fitting the data.
2) Selection of the "order" (number of functions) of the model.

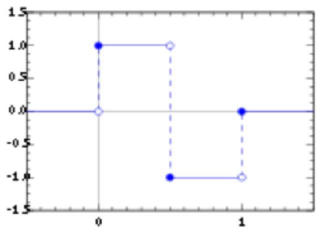
We can imagine a variety of possible models.
For example, the discrete cosine transform.

$$y_k(d) = \sum_{n=i}^{N} w_n \cos(2\pi \frac{d \cdot n}{N})$$

Haar Wavelets:

Haar A. Zur Theorie der orthogonalen Funktionensysteme, Mathematische Annalen, 69, pp 331–371, 1910.

The Haar wavelet is a difference of rectangular Windows.



$$h(t) = \begin{cases} 1 & \text{for } 0 \le t < 0.5 \\ -1 & \text{for } 0.5 \le t < 1 \\ 0 & \text{for } t < 0 \text{ and } t \ge 1 \end{cases}$$

The Haar wavelet may be shifted by d and scaled by s

$$h(t;s,d) = h(t/s - d)$$

Note that the Haar Wavelet is zero gain (zero sum).

$$G = \int_{-\infty}^{\infty} h(t)dt = 0$$

The Digital (discrete sampled) form of Haar wavelet is

$$h(n;d,k) = \begin{cases} 1 & \text{for d} \le n < d + k/2 \\ -1 & \text{for d} + \text{k/2} \le n < d + k \\ 0 & \text{for } n < d \text{ and } n \ge d + k \end{cases}$$

Haar wavelets can be used to define an orthogonal transform analogous to the Fourier basis. This can be used to define an orthogonal transform (the Walsh-Hadamard Transform). The basis is

$$H_0 = +1$$

$$H_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H_2 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

…

$$H_m = \frac{1}{\sqrt{2}} \begin{bmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{bmatrix}$$

Haar Functions, and the Walsh-Hadamard transform have been used in Functional Analysis and signal processing for nearly a century.

In the 1980s the Wavelet community re-baptized the Haar functions as "wavelets" and demonstrated that the Walsh-Hadamard transform is the simplest form of wavelet transform.

A 2-D form of Walsh-Hadamard transform may be defined using DoB features.
These can be calculated VERY fast using an algorithm known as Integral Images.