Intelligent Systems for Pattern Recognition

Federico Matteoni

A.A. 2021/22

Index

0.1 Introduction

Prof.s: Davide Bacciu and Antonio Carta

Objectives Train ML specialists capable of: designing novel learning models, developing pattern recognition applications using ML, developing intelligent agents using **Reinforcement Learning**.
We're referring to images and signals, but not limited to that: practical applications.
Focusing on challenging and complex data: **machine vision** (noisy, hard to interpret, semantically rich...) and **structured data** (relational information: sequences, trees, graphs...)
Natural Language Processing will be used as an example, but will not be the focus of this course.

Methodology-Oriented Outcomes Gain in-depth knowledge of advanced machine learning models, understanding the underlying theory. This gives the ability to read and understand and discuss research works in the field.

Application-Oriented Outcomes Learn to address modern pattern recognition applications, gain knowledge of ML, PR and RL libraries and be able to develop an application using ML and RL models.

Prerequisites Knowledge of ML fundamentals, mathematical tools for ML and Python.

0.2 Pattern Recognition

Automated recognition of meaningful patterns in noisy data.

Origins

Viola-Jones Algorithm Framework for face recognition. Sum pixel in white area and subtract those in the black portion. The VJ algorithm positions the masks on the image and combines the responses (training set of $\simeq$5k images with hand-aligned filters)

An historical view

1. Identification of distinguishing features of the object/entity (**feature detection**)
2. Extraction of features for the defining attributes (**feature extraction**)
3. Comparison with known patterns (**matching**)

Basically, lots of time spent hand-engineering the best data features.

A modern view Data is thrown into a neural network. A single stage process with a data crushing-and-munching neural network spitting out prediction, which encapsulates the three historical steps. But the time is now spent in fine-tuning the neural network.

The deep learning Lego Creating applications by putting together various combinations of CNN and LSTM modules.

0.2.1 Signals

Signals are time series: a sequence of measurements in time. Examples of sources are: medicine, finance, geology, IoT, biometrics...

Formalization A time series x is a sequence of measurements in time t

$$\mathrm{x} = x_0, \ldots, x_N$$

where x_t or $x(t)$ is the measurement at time t.

Observation can be at **irregular** time intervals.

We assume **weakly stationary** (or second-order stationary) data

$\forall\, t\ E[x_t] = \mu$
$\forall\, t\ \mathrm{Cov}(x_{t+\tau}, x_t) = \gamma_\tau$ with γ depending only on the lag τ

Goals

Description

Analysis: identify and describe dependencies in data

Prediction: forecast next values given information up to t

Control: adjust parameters of the generative process to make the time series fit a target

Key Methods

Time domain analysis: assesses how a signal changes over time (correlation, convolution, auto-regressive models)

Spectral domain analysis: assesses the distribution of the signal over a range of frequencies (Fourier analysis, wavelets)

0.2.2 Time Domain Analysis

Mean

$$\hat{\mu} = \frac{1}{N}\sum_{t=1}^{N} x_t$$

Can be used to subtract mean from values and "standardize" the two series.

Autocovariance For lag $-N \leq \tau \leq N$

$$\hat{\gamma}_x(\tau) = \frac{1}{N}\sum_{t=1}^{N-|\tau|}(x_{t+|\tau|} - \hat{\mu})(x_t - \hat{\mu})$$

Autocorrelation The correlation of a signal with itself.

$$\hat{\rho}_x(\tau) = \frac{\hat{\gamma}_x(\tau)}{\hat{\gamma}_x(0)}$$

We can compute this with every possible τ, finding the max/min which gives the τ where the autocorrelation is max/min, which means the lag where the signal starts repeating itself. The lags near zero typically dominates, so we want the maximum lag reasonably far from 0.

Autocorrelation plot

Cross-Correlation A measure of similarity of x_1 and x_2 as a function of a time lag τ

$$\phi_{\mathrm{x}_1\,\mathrm{x}_2}(\tau) = \sum_{t=\max\{0,\tau\}}^{\min\{(T_1-1+\tau),(T_2-1)\}} \mathrm{x}_1(t-\tau)\cdot \mathrm{x}_2(t)$$

Normalized cross-correlation Returns an amplitude independent value

$$\overline{\phi}_{\mathrm{x}_1\,\mathrm{x}:2}(\tau) = \frac{\phi_{\mathrm{x}_1\,\mathrm{x}:2}}{\texttt{todo}} \in [-1,+1]$$

With $\overline{\phi}_{\mathrm{x}_1\,\mathrm{x}:2}(\tau) = +1$ mean that the two time series have the exact same shape if aligned at time τ. Nearing -1 we get the maximum anticorrelation, same shape but opposite sign. Near 0 we get that the two signals are completely **linearly** uncorrelated.
Note that we measure **linear correlation**.

Cross correlation looks like the convolution

$$(f * g)[n] = \sum_{t=-M}^{M} f(n-t)g(t)$$

but we have a flipped sign ($n - t$ instead of $t - \tau$).
Cross-correlation is not symmetric, whereas convolution is ($f * g = g * f$).

Autoregressive Process A timeseries autoregressive process (AR) of order K is the linear system

$$x_t = \sum_{k=1}^{K} \alpha_k x_{t-k} + \epsilon_t$$

Autoregressive means x_t regresses on itself

$\alpha_k \Rightarrow$ linear coefficients $\mid |\alpha| < 1$

$\epsilon_t \Rightarrow$ sequence of independent and identically distributed values with mean 0 and fixed variance.

We look backward K steps, so limited memory.

ARMA Autoregressive with Moving Average process

$$x_t = \sum_{k=1}^{K} \alpha_k x_{t-k} + \sum_{q=1}^{Q} \beta_q \epsilon_{t-1} + \epsilon_t$$

With ϵ_t Random white noise (again)

The current time series values is the result of a regression on its past values plus a term that depends on a combination of stochastically uncorrelated information

Estimating Autoregressive Models Need to estimate: the values of the linear coefficients α_t and β_t and the order of the autoregressor K and Q
Estimation of the α, β is performed with the Levinson-Durbin Recursion (`levinson(x, K)` in matlab, and included in several Python modules).
The order is often estimated with a Bayesian model selection criterion, choosing the largest K and Q possible. E.g.: BIC, AIC...
The set of autoregressive parameters $\alpha_{i,1}, \ldots, \alpha_{i,K}$ fitted to a specific time series x_i is used to confront it with other time series. Same thing for β so we can use α for both sets.

Comparing time series by AR

timeseries clustering

novelty/anomaly detection

0.2.3 Spectral Domain Analysis

Analyze the time series in the frequency domain. Key idea: decomposing the time series into a linear combination of sines and cosines with random and uncorrelated coefficients. So a **regression on sinusoids** with Fourier analysis.

Fourier Transform Discrete Fourier Transform (DFT): transform a time series from the time domain to the frequency domain. Can be easily inverted back to the time domain.
Useful to handle periodicity in the time series: seasonal trends, cyclic processes...

Representing functions We know that, given an orthonormal system for E we can use linear combinations of the basis $\{e_1, \ldots\}$ to represent any function $f \in E$

$$\sum_{k=1}^{\infty} \langle f, e_k \rangle e_k$$

Given the orthonormal system

$$\{\frac{1}{\sqrt{2}}, \sin(x), \cos(x), \sin(2x), \cos(2x), \ldots\}$$

The linear combination above becomes the Fourier series `todo`

Representing function in Complex space Using $\cos(kx) - i\ \sin(kx) = e^{-ikx}$ with $i = \sqrt{-1}$ we can rewrite the Fourier series as

$$\sum_{k=-\infty}^{\infty} c_k e^{ikx}$$

on the orthonormal system

$$\{1, e^{ix}, e^{-ix}, e^{2ix}, e^{-2ix}, \ldots\}$$

Representing Discrete Time Series Consider x of length N and $x_n \in R$. Using the exponential formulation, the orthonormal system is finite, from e_0 to e_{N-1} each $\in C^N$
The n-th component of the k-th vector is

$$[e_k]n = e^{\frac{-2\pi ink}{2}}$$

Discrete Fourier Transform Given a time series x = $x_0, \ldots, x_{N-1}$ its DFT is the sequence

$$\text{Spectral domain} \quad X_k = \sum_{n=1}^{N-1} x_n e^{\frac{-2\pi ink}{N}} \quad \text{Time domain}$$

And can be inverted

$$x_k = \frac{1}{N} \sum_{k=1}^{N-1} X_k e^{\frac{2\pi ink}{N}}$$

Basic Spectral Quantities in SFT

Amplitude $A_k = |X_k| = \sqrt{Re^2(X_k) + Im^2(X_k)}$

Power $P_k = \frac{|X_k|^2}{N}$, more used in reality and under some conditions this is a reasonable estimate of

DFT in Action We use the DFT elements $X_1, \ldots, X_K$ as representation of the signal to train the predictor/classifier. This representation can reveal patterns that are not clear in the time domain.