Notes on ISPR

Vincenzo Gargano

February 24, 2023

Contents

Chapter 1

Lesson 1: Time series and brief to Fourier analysis

Generative and probabilistic models generally are used to learn probability distribution of our data and then drawing samples from them to generete new ones.

1.1 Pattern recognition

Some history: Duda and Hart are like founders, in that time there were't references to learning. Viola-Jones algorithm used for face recognition, based on filters, at the time they looked to 5k hand aligned examples!
Now we're doing essentially data 'matchers'. Usefulness of processing sequences is due to rescuing gradients.

Chapter 2

Lesson 2:

2.1 Time series

Signals that change in time from observation that are produced from a stochastic process. For example sensors, stock market, etc.
Sample can be irregularly space ot reguarly spaced, based on what we are working on.
One can change from irregularly sample to regularly sampled or we change the model.
A time series x is a sequence of measurements in time t. Time series analysis assume weakly stationary.
Expectation don't change with time : $\mathbb{E}[x_t] = \mu \forall t$
If you have a non stationary time series, Introduce a lag τ the Covariance is $\mathbb{E}[x_t x_{t+\tau}] = \gamma_\tau \forall t$ so the covariance between two observations doesn't change with type, without considering the lag

2.1.1 Goals for Time Series

What does means to analize time series? Is there any pattern that we can find across the time, con i use the past to predict the future?
Can i control a time series? Like robot movements in a continuous space.
Some common techniques are:

- **Baseline** Is the problem simple, complex... the performance and other.

- **Preprocessing** Is the data clean? Do we need to do some preprocessing?

Time analysis works with Correlation, Convolution and Autoregressive models.
Then we want to know the frequences **Spectral domain analysis**

2.1.2 Means and Autocovariance

Sample mean:

$$\bar{x} = \frac{1}{N}\sum_{t=1}^{N} x_t \tag{2.1}$$

Autocovariance is computed at specifig lag of the signal (our input), you take observations subtract means and then multiply them.
We can use autocovariance to compute autocorrelation, the autocovariance normalized by autocovariance without delay.

$$\rho_x(\tau) = \frac{\gamma_x(\tau)}{\gamma_x(0)}) \tag{2.2}$$

If two points are higly correlated, they are close to each other.
Autocorrelation gives us a picture of then a signal is repeating itself.
If we evaluate the autocorrelation at certain lags τ_i we can get an autocorrelation plot.
These will tells us the strong periodicity of the signal. Near the zero we have the trivial case, the autocorrelation is 1.

2.1.3 Two time series

Cross-correlation (discrete), a measure of two time series and how they are similar to each other take x_1 and x_2.

$$\phi_{x1,x2}(\tau) = \sum_{t=max0,\tau}^{N-1} x_1(t)x_2(t-\tau) \tag{2.3}$$

We can take the nomralized version of ϕ to get the cross-correlation, if this is 1 the series have exactly the same shape. Aligning them at the same τ,

at -1 you have the inverse one of the other, same shape but opposite sign (anticorrelated). At zero they are completely incorrelated (in a linear word), no linear dependancies, but there may be non-linear dependancies to take into account. Speaking linearly you can say that!

2.1.4 Convolution

Well cross correlation resemble really like convolution.

$$(f * g)[n] = \sum_{t=-M}^{M} f(n-t)g(t) \tag{2.4}$$

Convolution is a symmetric behaviour of the signal, if you cross-correlate two signal you get different results.

The derivative of the convolution has a nice property, derifing wrt f or g is the same, but one is an image and another is a filter, instead take the derivative of a filter is a much better and fast thing! So differential wrt the filters.

2.1.5 Autoregressive process

A time series AR of order K is the linear system, this is the classic autoregression but for time series.

$$x_t = \sum_{k=1}^{K} a_k x_{t-k} + \epsilon_t \tag{2.5}$$

We're working with the signal itself to compute future signal. Approximating it...
α_k are linear coefficient and ϵ sequence of iid values with mean 0 and fixed variance.
I've a vecotr α that contains the k coefficients and the error this will be computed in a more refined way because isn't really true that all is i.i.d, error made at previous time steps matters on the successive time steps. Then we weight the errors to get better estimation

$$\epsilon_t + \sum_{q=1}^{Q} \beta_q \epsilon_{t-q} \tag{2.6}$$

Model used here is ARMA, Autoregressive Moving Average.
Our parmeters we want to learn are α and β. This is linear model so that is the limit.
You need as well the autoregressor K (and Q), estimation of α is performed by Levison-Durbin recursion : matlab code is a = levinson(x, K);
Looking at the error the least the error the most complex is the model, the complexity matters and K represent how complex ths model will be. Order is estimated via Bayesian model selection criteria.
Let's call α^i set of autoregressive parameters fitter to a specific timeseries (dataset) x^i. Our data can contain different lenght time series, and after we get the set of parameter we get a transformation from irregular time series to a representation with same lenght. We can do timeseries clustering, novelty detection and we can even encode the time in a lower dimensional space, so we econd the time serie as a vector (flatten) and train with an MLP. It's an horrorific baseline, but it can be done...

2.2 Spectral Analysis

Another way to anlyze signal is to view frequencies.
Working with sound and music for example.
Key idea: decompose a time series into a linear combination of sinusoids. (and cosines) with random and uncorrelated coefficient, basically fourier analysis.
Time domain: regression on past values of time series Frequency: regression on sinusoids

2.2.1 Fourier Transform

Discrete Fourier Transform (DFT) is a linear transformation that maps a time series to a frequency series.
Can be easily inverted (inverse DFT) if you work in frequency domain is easy to handle periodicity of time series. For fourier lovers: This is an oversemplification.
Given an orthonormal system $e_1, e_2, ...$ for E we cna represent any function $f \in E$ as linear combination

$$f = \sum_{k=1} < f, e_k > e_k \tag{2.7}$$

With this orthonormal system:

$$\frac{1}{\sqrt{2}}, sin(x), cos(x), sin(2x), cos(2x), ... \tag{2.8}$$

The linear combination becomes the Fourier Series. We can as well go into the world of Complex values space to represent our function in a circle.

$$\hat{f}_k = \sum_{t=1}^{N} f(t) e^{-i2\pi kt/N} \tag{2.9}$$

Consider a discrete time series x_t with N samples. Using exponetiation formulation we get $e_0, e_1, ..., e_{N-1}$ vectors $e_k in C^N$. These are the root of unity, we want to compute c_k that is the integral (in discrete case is summation) of the elements in the orthonormal basis, so adding up every element.

$$x_k = \sum_{t=1}^{N} x_n e^{-i2\pi nk/N} \tag{2.10}$$

This is the DFT.
This is also the spectral representation of the signal.
So representing signal in frequency domain. The inverse transform is

$$x_n = \frac{1}{N} \sum_{k=1}^{N} x_k e^{i2\pi nk/N} \tag{2.11}$$

Be careful now we get the original time series based on the frequency. Complex conjugate.

2.2.2 Basic Spectal Quantities in DFT

We would like to measure relevance/strenght or contribution of target frequency bin k.

Amplitued and Power.
Power spectrume plot : we can plot the power of a signal that is a distribution it basically says where your signal distributed in a frequencies, maybe you don't want some frequences and you can get rid of what you don't want.
DFT in action, but in spectral domain to train a predictor, use the $X_1, ..., X_K$ as representing signal.
Represent in spectral domain can reveal patterns that are not clear in time domain.

2.2.3 Spectral Estimation

We have a lot of spectral analysis for example Spectral Centroid, spectrail weighted average frequency between bands b_1 and b_2. This is intresting to understand when you have music signals, for example what instrument are playing, centroids gives you what instrument is playing nice for genre classification, also you can use it for RNNs. If you fourier decompoe the behaviour of recurrent neuron (layer) in RNNs you can see the spectral centroid moves, that means that the network analize different things in each different layer. You can understand if you RNNs is behaviouring as a filter of a certain type.

2.2.4 Higher order moments

- Spread (variance-like)
- Kurtosis $(4^t hordermoment) Measure flatness or non-Gaussianity of spectrum around centroi$

For example you fourier decompose and then you go to see the Kurtosis, where has a peak you've a non-Gaussianity, you can use this to detect anomalies in time series.
Also Spectral Entropy if you plot this for different signal you can do pattern match of sounds representing the "peakness" of the signal.

2.2.5 Take home lesson

Old schoold pattern recognition on timeseries is still useful most of the time are linear methods, and fourier analysis allow us to identify key frequencies in the signal and these are powerful, but we can apply them to all sort of thing that have a frequence. Example if you have a graph in fourier domain by represeting it in adiacency matrix and then apply that to get an orthornomal basis so you can use fourier on Graphs!

Chapter 3

Lesson 3: Image processing - Descriptors

Today we talk about image and information and how can we find spatial and 'spectral' information about the images.
In images there will be a lot more of information.

3.1 Images are tensors

If we take an image like gray image this will be the greyscale matrix, when you color images you get 3 channels per image usually (R,G,B). Usually RGB isn't the best color space, there is the CIE-LUV color space a perceptual color space in which you can move "linearly", there are some libraries to change color spaces in python or with OpenCV.
What task we can do with images?

- Identification : Region of intrest identification (low level feature like lines, corners, blocks).
- Classification: More abstract and high level find a certain object, you can have different type of classification, like the most representative object in an image, but also to identify all the items.
- Segmentation: Coherence in the image based on the metric you use

- Semantic Segmentation: Based of the contents of the area, you can do a segmentation between (vegetation, sky, home, etc..). Usually you use a mixture of supervised and semi-supervised.
- Automated image capturing: from classification then you generate some sentences to describe the image.

How to represent visual information?

When we take an image we have raw signal not useful, but there are properties that we want it to have: Be informative, don't throw away useful information and maybe only throw redundant, also Invariance, for example if you move but take two picture, or take pictures with lights on or off, these variations are everywhere so we don't want to have descriptors to be weak towards these, some transformation like rotation that make the object different.
We want also it to be efficient, bigger the image more the pixels so... querying and indexing is difficult.
How to spot informative parts?
Represent distribution on the whole image, like colors, edges, corners: What are the main colors (discriminating bewteen the distirbution of color), distribution of edges.
Histograms are very good to represent distributions, a classical one could be number of pixels of a given color, but having histogram of a color what really tells us? You can transform the image.

What if i'm instrested to get info on certain part of the image, fro example background made of grass, my descriptor will be less noisy if it's local. Approaches local is more informative but things you need to reason is how do i find intresting points and how do i'm sure that extracting info on these is compliant with differences, different location, rotation and scaling of the item/thing that intrest us in the image.
Most of these are approached by convolution. A localized **descriptor** can be a patch of my image, i linearize it in a vector, so i have destoryed the information, this isn't invariant wrt intensity so no good descriptor, if i rotate image, i get different representation from my patch! These pixel based representation don't works. I want to represent my pixels exactly and i want that if i scramble the pixels i want. If i randomly permute all pixels in an image the histogram remains the same i had before scrambling! This is useful!
Represent patch by histograms describing properties, a simple approach is

to have histogram of pixel intensities on a subwindow but this isn't invariant enough, but we want invariant to illlumination (normalize it), scale (captured at multiple scale by simulating) and more.

3.1.1 SIFT

Scale Invariant Feature Transform, it's a descriptor that is invariant to scale Idea: Center the image patch on a pixel x, y of image I Represent the image at scale σ, control how close we look at an image. If we change sigma we blur or sharp the image, like a magnifing glass or mathematically speaking a gaussian. So we convolve the image with a Gaussian filter G, the larger the gaussian the blur is the image, the smaller is gaussian less are the other pixel that contributes!
We're in a discrete words so defined over descrete set, we create a grid then we run a gaussian on these grid that will become our patch, running it on our image will get us a convolution that with σ larger than 1 will blur instead with a small than 1 will sharpen the image. Basically on the ridges you can see very well when you apply the gaussian with low σ the black area will become very black and white will become more white and the border will be more visible. Next we can see how much we change from dark to brighter, we want the gradient of intensity in the patch, consider a magnitude m and orientation θ, using finite differences, so you're in an image and fix x and look at $y + 1, y - 1$ and the same with fixed y, with a bit of algebra and convolution we can simplify with this simple filter

$$G_x = [10 - 1] * L_\sigma(x, y) G_y = [10 - 1]^T * L_\sigma(x, y) \tag{3.1}$$

What i'm trying to do is understading the direction in which the intensity grows, so our image becomes a set of vectors that points towords that direction. Now that we got the gradient we can count the direction and move them to an histogram visualization, first we subdivide image in regions and we get the histograms that's basically a vector to count the different directions, we have 8 bins to contain our angles from a 4x4 patch, then we have our image made by 16 4x4 so you get 16x8. How do we get the invariance on the directions? We use an orientation gradient as main direction, so when we rotate the image we move this orientation, so basically you take a mean. So we use local gradients if one of the pixels from our camera has broken

pixels or noise we have a very black or very white pixel that broke the local gradient, so for this we can normalized to make it invariant in brightness. So basically SIFT descriptor is a 128 vector that contains the histogram of the gradient (directions) of the image. Sift is also a detector.

3.1.2 Fourier Analysis

Image are function returning intensity values $I(x, y)$ on the 2D plane spanned by x,y.

$$\mathcal{F}(f * g) = \mathcal{F}(f) * \mathcal{F}(g) \tag{3.2}$$

Transform convoutions in element-wise multiplication in Fourier domain, suppose an image $I * g = (F)^{-}1(\mathcal{F}(I) * \mathcal{F}(g))$ where F is the inverse Fourier transform, so we can see that the convolution is the inverse of the multiplication in the Fourier domain.

3.1.3 Image PR with DFT

Take original image and fourier transform it, then we fourier transform the thing we are looking for, we multiply and we do the inverse fourier then apply the threshold for high response.

3.1.4 Fourier Transform in Deep Learning

3.1.5 Practical issues with DFT on Images

DFT is symmetric in both direct

3.1.6 Take home message

These image representation are distribution based, very much about histograms, resistant to scrumbling, often gradients are based on intensity, intresting are where the gradients changes instantly, we like local descriptors because are very good.
Wavelet at a certain point: Now we spoke about analyze in time or frequency

domain, with wavelet we do both, we localize frequency analysis in different point of the image.