**PARTE 1 – MUSIC INFORMATION RETRIEVAL**

**LEZIONE 1 – Introduction to SOUND CLASSIFICATION (fino a slide 14)**

**MUSIC INFORMATION RETRIEVAL** **à** estrarre informazioni musicali.

* **SOUND ANALYSIS**: estimate some deterministic parameters of the sound, es. pitch, rhythm, melody ecc. NB: noi studiamo solo il **pitch.**
* **SOUND CLASSIFICATION**: for example infer a mood of a sound... happy, sad ecc. Non ci possiamo basare semplicemente sull’analisi dei parametri del suono ma abbiamo bisogno di tecniche più avanzate basate su machine learning, AI **à** **pattern-based systems**. Quindi andiamo ad estrarre high level information dal suono.

Riguardo ai sistemi di sound classification vedremo come:

* Window and segment a signal.
* Extract features from the sound che analizzate insieme ci possono portare ad una decisione sul fatto che per es. Il pezzo musicale è happy, sad ecc.
* Extract only the features that are meaningful per il problema di classificazione.

Perché studiamo sound analysis e sound classification?

Perché sono spesso il primo step per la creazione di altri sistemi, per l’analisi e risintesi del suono (per es. Voglio modificare i bpm del file ecc.), per creare database di suoni taggati secondo proprietà musicali e acustiche, per trascrivere i suoni in note.

**INIZIO SOUND ANALYSIS**

Tecniche per il rilevamento del pitch basate solo sull’analisi della waveform e sullo spettro del suono.

**PITCH DETECTOR (PD)**: algoritmo software o hardware che prende un segnale audio in ingresso e cerca di determinare in output il fundamental pitch period del segnale.

PD funziona soltanto per alcuni suoni; per esempio non va bene per trovare il pitch di percussive sounds, impulses o suoni complessi composti da più strumenti magari. Posso usare Pds solo per single pitched sources.

Casi in cui i PDs hanno difficoltà a determinare il pitch:

1. *Attack transients* **à** perché le waveform sono caotiche e instabili durante l’attack (attack: very first part of the sound).
2. *Low Frequencies* **à** perché i PDs lavorano in window e per ogni window analizzano il pitch. Se il suono è a una frequenza molto bassa posso osservare magari solo una singola oscillazione in quella window... Per performare una stima del pitch i PDs hanno bisogno almeno di 2-3 cicli di oscillazione. **à** stima inaccurata o assente.
3. *High frequencies* **à** il periodo della waveform è rappresentato da pochissimi samples, 3-4 magari. Ed è difficile stimare il pitch di un suono usando solo 3-4 samples. **à** stima inaccurata.
4. *Myopic pitch tracking* **à** i PDs registrano dettagli irrilevanti che sono stati prodotti non intenzionalmente. Infatti, i PDs analizzano il suono su intervalli da 20 a 50 ms, ma la percezione del pitch nell’uomo è basata su intervalli di tempo maggiori.
5. *Acoustical ambience* **à** la rilevazione del pitch è influenzata dall’ambiente in cui si registra lo strumento o la voce.

**Categorie di PDs**

1. *Algoritmi time-domain* **à** per trovare patterns che si ripetono nella waveform.
2. *Tecniche basate sull’autocorrelazione* **à** per trovare patterns che si ripetono nel segnale, indicatori di periodicità.
3. *Tecniche di filtro adattivo*
4. *Frequency domain*
5. *Modelli dell’orecchio umano*

**NB1**: fundamental frequency = pitch.

**NB2**: fundamental frequency = frequenza più piccola di una waveform.

1. **Algoritmi time-domain**

* *Zero-crossing PDs*: vado a misurare l’intervallo tra 2 zero-crossing successivi della waveform; in questo modo riesco a rilevare il pitch della waveform, cioè considero l’inverso dell’intervallo di tempo tra 2 crossing e trovo appunto la frequenza di crossing, cioè la stima del pitch.

**NB**: Zero-crossing **à** punto in cui cambia il segno di una funzione. Il segnale attraversa lo 0- axis.

* *Filter-bank zero crossing* (per minimizzare l’influenza del rumore): faccio lo zero-crossing ma prima il segnale passa in un filter-bank, facendo passare solo una specifica banda del segnale. Per esempio se ho del rumore (che ha un’ampiezza di banda grande) lo posso filtrare per favorire solo il segnale (che tipicamente è concentrato in una banda ristretta). Quindi l’algoritmo controlla l’ampiezza (massima variazione del segnale) dei segnali che escono dagli output di ogni filtro del bank e esegue il rilevamento zero-crossing sui 2 filtri che mi danno un’energia del segnale maggiore.
* *EGG* *o laryngograph*: si indossa una fascia al collo con degli elettrodi che misurano gli impulsi emessi dalle corde vocali.

**2. Autocorrelazione** **à** ho un segnale: considero il segnale all’istante t e lo stesso segnale all’istante t + τ per verificare quanto il segnale si somigli (si autocorreli) all’avanzare del tempo.

Se il segnale varia lentamente nel tempo, l’autocorrelazione è positiva. Se invece varia rapidamente, è prossima allo zero.

Se ho un segnale con periodo L, nell’autocorrelazione troverai massimi ai lags (ritardi), L, 2L, 3L ecc. Quindi L è il minimo divisore per cui posso trovare dei massimi. (??)

L’idea è ottenere una stima robusta del pitch osservando tutti i maxima nell’autocorrelazione e trovare il miminum divider. (??)

**3. Tecniche di filtro adattivo**

* *Filtro passabanda adattivo*: immetto il segnale di ingresso in un filtro passabanda a banda stretta (very narrow) regolabile. Il filtro modifica la sua frequenza centrale per minimizzare la differenza tra segnale in ingresso x(n) e segnale in uscita y(n) del filtro passabanda. In pratica se il filtro passabanda è centrato nella frequenza centrale del segnale x(n), allora l’output y(n) sarà il massimo del segnale x(n), dunque il test di convergenza che confronta l’energia di x(n) con quella di y(n) troverà un minimo, cioè la distanza tra i 2 segnali è minima. Quindi la central frequency del filtro passabanda coincide con la frequenza fondamentale del segnale, cioè con il pitch.
* Filtro a pettine ottimo: cerco un filtro a pettine che minimizza il segnale in uscita. Comb filter **à** filtra sia la fundamental frequency che le harmonics. (??)

**4. Frequency domain pitch detectors** **à** cercano picchi nello spettro che corrispondono a frequenze rilevanti. Dopo aver trovato questi picchi, il PD deve decidere quali di queste frequenze sono fondamentali e quali sono armoniche o parziali. Problema tipico: l’orecchio umano divide le frequenze in una scala logaritmica, perciò è molto sensibile nello stimare il pitch alle low frequencies, mentre i computer tipicamente lavorano con una scala di frequenze lineare, perciò non sono così sensibili nello stimare il pitch alle low frequencies. Perciò c’è un problema di resolution che limita l’accuracy del pitch detector basato sul frequency domain.

**5.**  **Modelli orecchio umano à** i PDs in questo caso emulano le funzioni di ogni componente della dell’orecchio umano e dunque il risultato del suono processato dall’orecchio esterno e medio, dalla coclea e dal sistema nervoso centrale.

INPUT SIGNAL **à** OUTER AND MIDDLE EAR **à** COCHLEA (inner ear) **à** CENTRAL NERVOUS SYSTEM

*Outer ear & middle ear* --> component that performs a *preliminary spectrum shaping of the signal*, so of the envelope of the signal (obviously, in the frequency domain).

*Cochlea* (responsible for forming a band suddivision of the incoming signal; actually the cochlea can be modelled has a filter cascade with half-wave rectifiers) --> component that performs: *decomposition into frequency bands*, *neural transduction* (which performs the half-wave rectifier) and *spike generation* (for each half-wave in the signal generates a spike to be passed to the brain).

**NB**: rectifier (raddrizzatore) --> trasforma il segnale sinusoidale in entrata (cioè un segnale alternato) in un segnale unidirezionale (cioè o sempre positivo o sempre negativo).

*Central nervous system* --> component that detects the intervals between consecutive spikes (*interval detection*).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**LEZIONE 2 – Introduction to SOUND CLASSIFICATION (fino a slide 41)**

Last tool available for sound pitch estimation:

**CEPSTRUM ANALYSIS**

*Idea*: estimate the excitation signal (ex. Vocal chords) just having available the microphone that is placed in front of the mouth of the singer.

Il suo nome deriva dal capovolgimento delle prime quattro lettere della parola "spectrum".

Attualmente è uno strumento molto efficace per discriminare la voce umana nell'informatica musicale.

*Goal dell’analisi cepstrum* **à** estrarre l’excitation signal dal segnale osservato.

In che modo posso estrarlo? Devo trasformare la moltiplicazione in una somma di logaritmi e poi posso filtrare l’effetto di log H(f) (la componente che non mi interessa).

**NB**: siamo nel frequency domain (nell’equazione con i logaritmi), ma di solito applichiamo il filtering su dei segnali che hanno un’additive relationship nel time domain. Quindi visto che siamo nel frequency domain dobbiamo tornare indietro al time domain, calcolando l’IFFT, e poi applicare il filtro (un’High-pass filter perché sono interessato solo alla componente di Y a variazione veloce, cioè a X).

Poi l’High-pass filter just selects components dall’output dell’IFFT above a given time-lag.

Tutto ciò viene eseguito dopo l’acquisizione del segnale e non quindi in real-time.

**NB**: la rappresentazione logaritmica non cambia il comportamento di H(f) e X(f). Infatti, le variazioni di queste due funzioni si riflettono allo stesso modo nelle loro controparti logaritmiche.

Excitation signal X(f): la vibrazione originale del suono e il pitch (vocal chords signal) **à** componente di Y a variazione veloce.

Resonance effect H(f) (funzione di trasferimento) (filtering effect of the vocal tract): attuata da una struttura risonante: laringe (che contiene il timbro della voce, cioè è in pratica l’impronta digitale di una voce) o corpo dello strumento musicale **à** componente di Y a variazione lenta.

NB: il segnale osservato può essere considerato come la convoluzione tra il segnale di eccitazione e la risposta impulsiva del filtro del tratto vocale.

Segnale osservato: y(t)=(h\*x)(t) --> FFT --> Y(f)=H(f)X(f).

**FINE SOUND ANALYSIS**

**INIZIO SOUND CLASSIFICATION**

Non vogliamo semplicemente analizzare il segnale, ma vogliamo cercare di arrivare ad una comprensione più profonda di esso (per estrarre caratteristiche che sono più meaningful for the humans), dunque si combinano tools di low-level signal processing con tecniche software di intelligenza artificiale.

Ci sono due gruppi di sistemi di sound classification: quelli che cercano di simulare le listening skills dei trained human listeners, e quelli che *non emulano gli human listeners*. **Noi ci occuperemo solo del secondo gruppo.**

*Pattern-based classification*: estraggo descriptors sintetici (chiamati features) dalla waveform o dallo spettro... Esempi di descriptors sono: energy della waveform, centroide spettrale ecc.

Tutti questi descrittori formeranno un point nel multi-dimensional space, which is called feature space.

Raggruppo i descriptor in un vettore e poi li rappresento nel feature-space (nel caso N=2 ho una feature in ascissa e una in ordinata).

Then for each istance (portion --> segment or window) of the signal we obtain a point in this multi-dimensional space by the juxtaposition of all the features.

Example: we want to discriminate in the signal the presence of noise or the presence of voice. If the features have been chosen properly, we’ll expect that segments coming from voice will cluster in a specific area of this multi-dimensional space, and the segments coming from noise in another area.

Perciò applichiamo una sorta clustering operation in modo da ottenere una rappresentazione “geografica” of noise and voice signals. Quindi quando abbiamo un sample che non è stato classificato, semplicemente guardando alla sua posizione nello spazio we are able to identify whether it is noise or voice.

**NB1**: the circles in the graphical example represent istances of the signal.

**NB2**: centroide spettrale: è il centro di massa dello spettro, cioè individua il punto medio della distribuzione di energia di un suono. L’energia di un segnale è data dall’integrale del modulo al quadrato del segnale, con estremi da –inf a +inf.

Il segnale contiene diversi eventi audio che devono essere classificati; se le features scelte sono rilevanti per il problema, i punti nel feature space si raggruppano in modo tale che ogni cluster nel feature space può essere assegnato ad una classe audio.

**GOAL of SOUND CLASSIFICATION systems**: assign an istance of a signal to a specific class.

Per raggiungere questo goal **à** è necessario un training stage (per imparare come lo spazio è suddiviso/organizzato) **à** inserisco un set di istances nel training system per stimare cluster all’interno del feature space.

**NB**: ci riferiremo a sistemi di training supervisionati (supervisionati perché il training set è labelled, quindi per ogni istance in the training dataset, we know in advance which class this istance belongs to).

Dopo il training stage ho una fase di test in cui il system can be run for the classification.

**ESEMPI di SOUND CLASSIFICATION**: music transcription automatica (es. il pc ascolta la musica e la trascrive automaticamente), sistemi di speech recognition, sistemi di sorveglianza audio-based, music recommendation.

*Criteri per avere una successful classification*:

* Scegliere un set di features rilevanti **à** i cluster devono essere distinti tra loro e ogni punto nel cluster deve essere vicino al centroide del cluster.
* Scegliere un opportuno sistema di classificazione
* Scegliere un training set rilevante

**ARCHITETTURA DEI SISTEMI DI SOUND CLASSIFICATION (traditional classification)**

1. WINDOWING/SEGMENTATION del segnale (divide the signal in small portions)
2. DE-NOISING (non trattiamo questo blocco)
3. FEATURE EXTRACTION (from each portion we extract the features)

Dopo questi tre step ho un branching in the diagram: o andiamo al training stage o al classification stage. During the training we learn how the features are organized, so how the space is clustered and then in the classification is given an unseen istance and we decide for its class using an appropriate decision function.

**Windowing** **à** vado a frammentare il segnale in più windows consecutive che sono distanziate in modo regolare, cioè hanno la stessa lunghezza (devo vedere come un segnale viene windowed!!).

**Segmentation** **à** vado a estrarre frammenti di segnale che sono consistenti da un punto di vista semantico. Il segnale è frammentato in finestre che non sono distanziate in modo regolare (quindi non è detto che abbiano la stessa lunghezza) ma distanziate in base a questa proprietà di uniformità.

La segmentation è utile quando il costo computazionale è un problema: ci permette di evitare di classificare porzioni di segnale che non sono rilevanti per il nostro problema (es: assenza di un parlato in un sistema di riconoscimento di un parlato appunto).

We can categorize the features according to *different categorization systems*, one of the most used ones is the **MPEG-7**.

**FEATURE EXTRACTION** **à** categorie di features che si possono estrarre:

* *Basic descriptors* (work on the wave in the time domain) & basic spectral descriptors (work in the frequency domain)
* Basic signal parameters (we won’t study these parameters)
* *Temporal timbral descriptors* (they aim at characterize the signal from its timber, for example finding the length of the Attack phase or maybe the Sustain phase and so on, for example for discriminating from a drum sound and a piano sound) & *spectral timbral descriptors* (aim at analyzing the sound as an example from the formants point of view)

**NB1**: we know that every musical instrument has a specific timber in terms of Attack, Decay, Release and Sustain.

**NB2**: definition of formants **à** determinate bande di frequenza, tipiche di una sorgente sonora (strumento musicale acustico: cioè nè elettrico e nè elettronico, voce umana) che ne caratterizzano fortemente il timbro.

* Spectral basis representations (we won’t study these parameters)
* Gli altri descriptors non-standard non possono rientrare in nessuna di queste classi.

Notazione per i descriptor a pag. 32.

**NB** (important): we will use t for discrete signals!!

**BASIC DESCRIPTORS**

* *AUDIO WAVEFORM (AWF)*: nell’esempio considera come time frame 1/100 di secondo (perché andiamo a considerare il tempo discreto!). Dunque, per ogni frame vado a considerare l’ampiezza massima (maxRange) e minima (minRange) del segnale. L’AWF sarà data dalla serie di tutte queste coppie (minRange, maxRange). In pratica l’AWF non è altro che il contorno del segnale (l’envelope della waveform).
* *AUDIO POWER (AP)*: for each frame, estrae la potenza media del segnale. In pratica vado a considerare la media quadratica dell’ampiezza del segnale per ogni frame (def. di potenza media).

**BASIC SPECTRAL DESCRIPTORS**

* *AUDIO SPECTRUM ENVELOPE (ASE)*: idea **à** estrarre un’envelope of the spectrum, cioè vogliamo “riassumere” lo spettro del segnale (che ha many samples) with just a few numbers. Quindi dividiamo il power spectrum in sub-bands and for each sub-band we sum all the samples in the same sub-band. Alla fine, abbiamo una rappresentazione che ha lo stesso numero di samples as the number of sub-bands. È un log-frequency spectrum che può essere usato per generare uno spettrogramma ridotto del segnale originale. Come lo ottengo? Devo sommare l’energia del power spectrum dentro una serie di bande di frequenza. Le bande sono distribuite in modo logaritmico (base 2) tra 2 frequenze estremità che sono loEdge e hiEdge.

Formula pag. 36 ed esempio pag. 37.

**NB1**: la band suddivision è fatta in un non-uniform way as the bands in the low frequency range are more closely spaced (sono più ravvicinate) than the high frequency range. Questo perché il nostro brain è più sensibile alle low frequencies piuttosto che alle high frequencies, perciò questo deve essere considerato.

**NB2**: r è un numero (da 1.8 a 16) che controlla la resolution dell’ASE; smaller is r, higher is the resolution, perché è minore la distanza tar il lower e l’upper limit di ogni banda.

Passband **à** mi riferisco alla posizione della band, quindi alla sua central frequency.

* *AUDIO SPECTRUM CENTROID (ASC)*: idea **à** va a considerare il log- frequency power spectrum come una probability density function. Dunque, computa il centro di massa di questa function. Il grafico dell’ASC drops when we have noise.

**NB1**: prima di computare il centro di massa andiamo a considerare un power spectrum modificato in which all the frequencies below K\_low (che è un bin) are summed together, quindi si modifica anche il frequency-axis **à** the first element of this axis is at 31.25 Hz = 62.5/2 Hz.

**NB2**: in slide 39 the noise in concentraded mostly in the low frequency band. Actually, when there is a pause, the spectrogram exhibits a magnitude different from 0 only at low frequencies.

* *AUDIO SPECTRUM SPREAD (ASS)*: considera ancora una volta il power spectrum (quello slightly modified) come una probability density function, ma in questo caso computa lo spread, che è la square root della variance. Per un dato frame del segnale, l’ASS rappresenta lo scarto quadratico medio (radice quadrata della varianza) dello spettro rispetto al suo centroide ASC.

**NB**: when we have just noise, the audio spectrum spread increases due to the fact that we have just noise, and noise we know is typically wide-band, quindi abbiamo uno spread più grande.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**LEZIONE 3 - Introduction to SOUND CLASSIFICATION (fino a slide 71)**

**NB**: LLD penso voglia dire Low Level descriptor.

**AUDIO HARMONICITY** **à** we analyze how the sound can be considered armonic; this descriptor provides 2 misure delle proprietà armoniche dello spettro:

* *Harmonic ratio (HR)*: the ratio of harmonic power to total power of the signal

Starting point: compute the normalized autocorrelation function, given the signal s\_l(t) (s\_l(t) is the l-th frame from s(t)).

So we look for the sample at which we have the maximum of the normalized autocorrelation in a given lag window that goes from M\_0 to M.

**NB**: more power is the harmonic component of the signal, higher will be HR.

* *ULH*: the frequency oltre la quale lo spettro non può essere considerato armonico **à** u compute the ratio between the spectrum of the signal and the spectrum of a modified signal in which u remove the armonic component. Per rimuovere l’harmonic component devo sottrarre al segnale s\_l(t) a scaled and delayed version of itself.

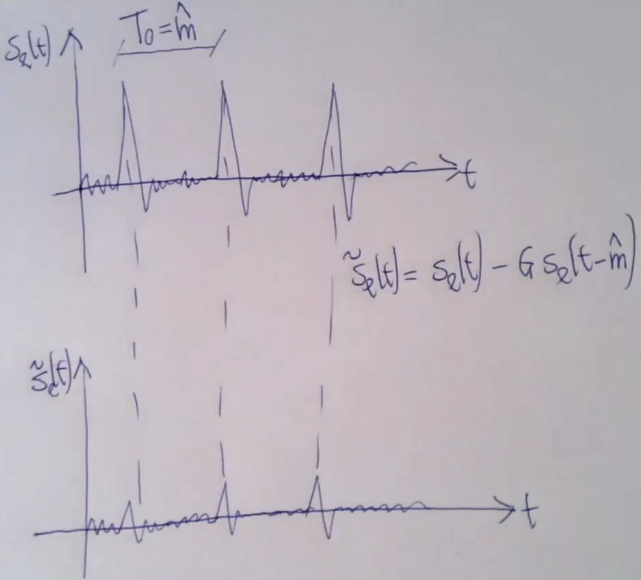
m hat **à** estimated fundamental period of the signal.

G **à** scaling factor (stimiamo G per minimizzare l’energy di s\_l tilde); è l’optimal gain of the comb filter (vedi formula slide 46).

s\_l tilde **à** noisy component del segnale; poi da questa computo lo spettro e faccio la ratio.

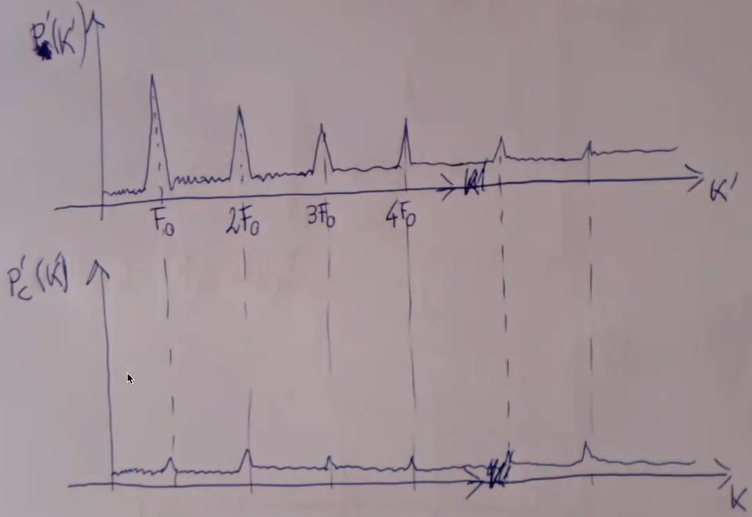
The signal is comb-filtered in order to filter-out the harmonic effect.

**Esempio grafico**:

s\_l(t) is an armonic signal, so it is characterized by some spikes, alternated with some areas where the amplitude is much less important. The distance between 2 consecutive peaks is the fundamental period T0. What we are doing with s tilde is removing the energy associated to all the periodic components of the original signal, cause the delay is equal to the fundamental period of the signaland therefore the energy associated to the periodic components will be minimized, in particular in correspondance to the spikes. Basically we are subtracting from the signal the armonic component.

Then starting from s\_l and s\_l tilde we have to compute P’(k’) e P’\_c(k’), che sono i power spectra dei 2 segnali (\_c means comb-filtered).

**NB**: *Definition of FREQUENCY BIN* **à** is an interval of frequencies, obtained when we discretize frequencies. So for example our continuous frequencies become N discrete bins.



F0 is the fundamental period. Then, the first armonic is 2F0, second is 3F0 ecc. Formula R(K) **à** vedi slide 47. Then it will happen that R(k) will be equal to 0.5 for the frequency bin k=k\_lim. The frequency corresponding to the frequency bin k\_lim is the frequency of the upper limit, denoted by f\_ULH. Poi per ottenere ULH faccio il log in base 2 di f\_ULH/1000. Usiamo k’ perché devi considerare il modified spectrum della slide 38.

Klow is the number of bins that all together can contribute to the lowest element in the power spectrum (perché devi considerare che nel power spectrum modificato sono andato a sommare tutte le potenze dal bin 0 al bin K\_low in unico coefficiente). Quindi, il numero totale di frequency bins non è N\_ft/2 (che sarebbe il n° dei frequency bins fino al bin della Nyquist component) ma è N\_ft/2 - K\_low.

NO Audio Fundamental Frequency

**NB**: *frequenza di NYQUIST* **à** indicata con F\_s/2, F\_s è la frequenza di campionamento (sampling rate, cioè n° di samples al secondo).

Nelle sommatorie dove compare N\_ft/2 - K\_low, N\_ft/2 va inteso come il frequency bin associated at the Nyquist component, cioè k\_Nyq.

**OTHER NON MPEG-7 DESCRIPTORS**

* Zero-Crossing Rate **à** it counts how many times the waveform crosses the 0-axis, computando la differenza tra il segno di s(t) e il segno di s(t-1). (51)
* Spectral Roll-off Frequency **à** frequency below which 85% of the accumulated spectrum magnitude is concentrated NB: in this case we are working with the spectrum, not the power spectrum. The frequency bin K\_roll (converted to corresponding frequency) is known as the spectral Roll-off frequency.
* Spectral Flux **à** average variation of the signal amplitude between 2 adjacent frames. È la differenza al quadrato tra i log di 2 successive spectral distributions (di 2 frames adiacenti del segnale) summed over all the possible frequencies bins and summed for all the possible time-frames. It’s useful for signals that are segmented, cause the spectral flux works segment by segment. We are taking the same frequency bin of 2 consecutive frames and we are computing the difference of the logarithms of their spectra. NB: the delta is used just to avoid computing the logarithm of 0.

**TEMPORAL TIMBRAL DESCRIPTORS** **(classified within the MPEG-7 descriptors)**

Extracted from the signal envelope in the time domain **à** they describe the timbre (whatever characterize the signal as a perceptional view).

Col *timbro* vado a distinguere due suoni che sono comunque uguali in loudness, pitch e subjective duration.

*Signal envelope* **à** describes the energy change of the signal **à** it’s equivalent to the ADSR model of a musical sound **à** Attack, Decay, Sustain, Release. (vedi grafico slide 54)

**Temporal timbral descriptors**:

* *Log attack time (LAT)* **à** time that goes from T\_start to T\_stop; it’s the time the sound envelope takes per raggiungere l’ampiezza massima del segnale from a minimum treshold time. Per treshold time si intende tempo limite (inferiore), e nel nostro caso è T\_start. In T\_stop we have te maximum envelope of the signal. È il logaritmo della duration, cioè log(T\_stop-T\_start).
* *Temporal centroid* **à** tempo medio calcolato rispetto alla signal envelope Env (vedi slide 56).

**NB**: un modo per computare l’Env(l) **à** vedi slide 56 **à** it’s the square root of the average power of the signal over a given time window.

**Final descriptor that we will describe**:

* *Mel Frequency Cepstrum Coefficients* (57-60) **à** used for representing the human voice and musical signals. It’s the mostly used as an example in the speech recognition systems. Mel frequency because we apply a non-linear modification of the frequency axis, according to the mel-frequency scale, so that after the modification the signal is more resemblant (somigliante) to how the signal is processed by the human ear.

**NB**: We’ll use a non-linear mel-frequency scale; this scale approximates the behaviour of the auditory system, in terms of frequency resolution (our auditory system exhibits a better frequency resolution in the low frequencies rather than the higher frequencies).

Per convertire la frequenza da Hz a mel, vedi formula slide 58 (il mel è una unit of pitch **à** subjective impression of frequency).

The mel scale is a scale of pitches judged by listeners to be equally spaced one from the others.

In order to process the signal in the MFCCs, we have to derive triangular filters which are equally spaced on the mel scale. You have pass-band triangular filters.

**Scheme for the extraction of MFCC’s** **à** slide 60 (*slide 59 saltata*)

1. Sample s(t)
2. Window it
3. Take the FFT of the signal
4. Apply the Mel Filter Bank
5. Take the log of the output energy of the Mel Filter Bank
6. Finally apply the DCT: Discrete Cosine Transform (it’s similar to the IFFT, but we apply the DCT cause we know that we have real data numbers; actually DCT just works with real data)
7. We obtain the MFCC coefficients

Typically, after computing the MFCCs, u compute the derivative of the MFCCs and the second order derivative (acceleration) because these estimations bring some information on the signal.

**NB**: the role of the log operation in the MFCCs is to convert the multiplicative relationship between slowly and rapidly varying components of the spectrum into an additive one.

**SOUND CLASSIFICATION: TRAINING AND TESTING**

Once the audio features have been extracted from the signals, they are fed (vengono fornite) to the classifiers. If the chosen features are relevant for the problem at hand, they will cluster in the feature space in different classes. Each cluster refers to a single type of acoustic event (class).

**TRAINING (supervised)** **à** given a “labelled” dataset (we know in advance which class each element belongs to), estimate the parameters of one or more functions that partition the feature space into different portions, each belonging to a specific audio event. So we want to know how the feature space is divided for different audio events.

**TESTING** **à** given an unlabelled data and the parameters of the above function, we want to estimate the class it belongs to.

**3 STRATEGIES FOR SOUND CLASSIFICATION**

They differ according to the specific parametric function used to cluster the feature space:

1. **k Nearest Neighbors (kNN)** **à** decides the class of an unknown (unlabelled) point in the feature space based on the class of the k Nearest Neighbors (for the explanation of the strategy see slide 65).

The choice of the distance function is critical for the classification (formulas slide 66):

For continuous features:

* *Euclidean distance* (just the distance between 2 points)
* *Manhattan distance*
* *Minkowski distance* (è una generalizzazione delle prime 2)

For non-continuous features:

* *Hamming distance* (numbers of different elements between 2 signals, expressed with numbers)

How to solve the problem with uneven (ineguale, irregolare) class distribution? (for example, when one of the classes is represented more than the other ones) **à** we can partition the feature space before perfoming the classification, so in the training phase. In this way, every partition, every area, belongs to a different class (see image in slide 67). Then the test stage is accomplished by finding the region in which the point falls.

1. **Gaussian Mixture Models (GMM)** **à** the function on which we base our decision is a mixture of gaussian probability density functions (pdf’s) **à** this belongs to a wider class that’s the Bayesian classification **à** based on probability theory **à** we want to find the class with the highest probability of being the generator of the observed feature vector, so that the error risk in getting a wrong classification is minimized. (see notation slide 69)

The idea is to find the class ω\_k that most probably generated the feature vector x under observation. For doing this we need the probability of ω\_k given x, so the probabilty that ω\_k generated the observed vector x. Once the PDFs are available or have been estimated, the classification finds the class ω\_k that maximizes that probability.

**Bayes theorem** **à** lo usiamo perché la probabilità p(ω\_k|x) non è nota a priori e non la possiamo stimare direttamente.

Il problema si riconduce a stimare p(x|ω\_k) e p(ω\_k).

* p(ω\_k) **à** a-priori probability (perché non dipende da x) della k-esima classe. La possiamo stimare o assumendo una distribuzione uniforme (all the classes are equally probable) oppure stimarla basandoci sui corpora (so we are given before a very huge training dataset and we annotate this training dataset finding the probability distributions of all the possible classes). This last approach is important for applications such as speech recognition.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**LEZIONE 4 - Introduction to SOUND CLASSIFICATION (fino a slide 95)**

* p(x|ω\_k) **à** for each class ω\_k we have to estimate this probability. For doing this we’ll resort (ricorreremo) to the training dataset.

**Gaussian Mixture Models** (72, 73) **à** model of a probability density function which is a mixture of gaussian probability density functions (vedi slide 72-73).

λ collects all the 3 parameters that control the GMM; they are c\_m (coefficient), Σ\_m (matrix), μ\_m (vector). To estimate these components we have training datasets.

**Training of GMM’s** **à** we find the parameters that maximize the likelihood of the dataset **à** **Expectation Maximization (EM)**.

It is based on an expectation step (where we compute an expectation of the likelihood of the features vectors x\_i) and a maximization step (where we compute the parameters that maximize the expected log likelihood found in the E step).

We continuously iterate between these 2 steps until we reach some convergence condition **à** for example the log likelihood criterion (slide 78)

M **à** given the likelihood, we update the estimate of the distribution (slide 77)

E **à** given the parameters, we update the estimation of the likelihood (slide 77)

**NB**: In the EM we know in advance the number of classes (M: number of classes); if we don’t know M, we can use other techniques that estimate also the M that best optimizes the distribution, so the M that guarantees the maximum log likelihood value l(λ).

**Testing of GMM’s à** we just estimate the probability p(x|ω\_k), but we already know everything: we just apply the Bayes theorem.

Before performing classification u have to normalize the features! And keep track of this normalization adopted.

1. **Support Vector Machines (SVM)** **à** the feature space is partitioned by hyperplanes.

Everything is deterministic; it’s a binary classifier (in nature) that learns the boundary between items belonging to 2 different classes. **à** see example slide 79 (the idea is to find an hyperplane that separates the 2 classes and maximizes the distance between the 2 classes).

**Definitions for SVM***: support vectors* (they are the points circled that lie in the boundaries of the 2 classes), *margin* (z\_2 in the example), *(non) separable SVMs* (there are some points of the class black circles that go beyond the boundary and could eventually enter in the region of the crosses or vice versa), *(non) linear SVM* (we can something that divides the 2 regions, but it’s not a linear subspace... for example it could be a curve or something else).

Hyperplane **à** linear subspace whose dimension is one less than the space dimension.

Mathematical def. of an hyperplane --> slide 81

Only the support vectors determine the direction w of the hyperplane (è il coefficiente angolare della retta); w\_0 sets the distance of the hyperplane from the origin of the axis (è la quota q di una retta in pratica).

**Training of SVM à** we want to find the hyperplane that separates the 2 classes and we want to maximize the margin (slide 82).

**NB**: g(x)=0 sull’hyperplane, =1 nel boundary superiore, =-1 nel boundary inferiore, >1 sopra il boundary superiore, <1 sotto il boundary inferiore. The margin is 2/||w||.

So if we want to maximize the margin, we have to maximize 2/||w||, but this quantity is strongly nonlinear. Therefore, instead of working with this function, we work with an equivalent function, which has to minimized. The new function is J(w)=1/2||w||^2. So we are dealing with a constraint minimization problem (slide 83).

The optimal hyperplane is a linear combination of some of the feature vectors.

**Decision function** (85) **à** given a generic point x, we want to infer the class it belongs to (given w and w\_0) **à** we just see the value of g(x): if >=1 class 1, if <=1 class 2, if between –1 and 1 no decision is taken.

**Non-linear SVM** (86) **à** we operate a transformation (using different functions, there are many of them) on the feature space, which transforms the non-linear boundary into a linear one, so that we can find a hyperplane that separates the 2 classes.

**Non-separable SVM** (87) **à** we introduce a penalty factor, that allows some points to overcome the boundary, but will give some penalty on the optimization, so the value of the cost function will be increased. In this case, at the end, the hyperplane that optimizes the cost function will not have too many points that go beyond the boundary.

**NB**: SVM is a binary classifier, cause we have only 2 classes.

**FEATURE SELECTION**

Find a subset of features which can guarantee best classification results, cause when a large number of features is present, we could have problems of overfitting (many features provide information that are very similar) and/or computational cost.

Feature selection is positioned between feature extraction and classification.

2 possible approaches for feature selection **à** Filter methods and wrapper methods (studiamo solo i filter methods).

* **Filter methods** **à** the algorithm we apply for the feature selection is indipendent from the classifier used.
* **Wrapper methods** **à** the algorithm works by analyzing the accuracy of the classifier.

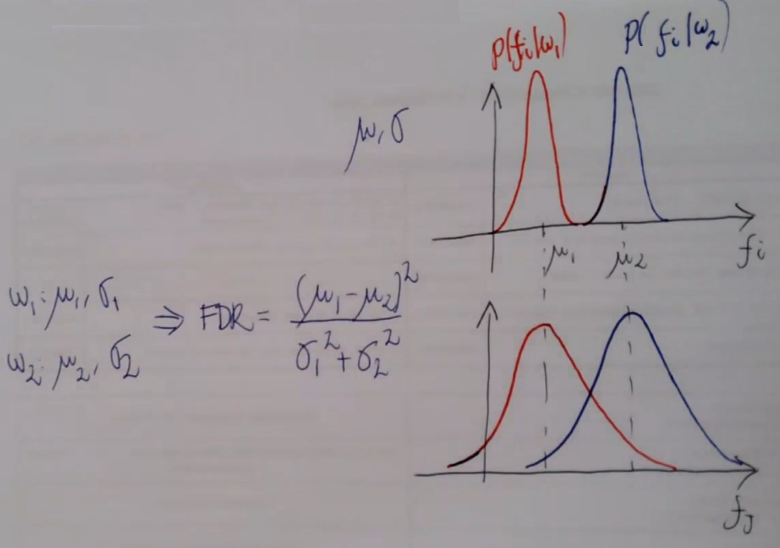
There are *2 kinds of feature selection*: **scalar** and **vectorial** (we’ll analyze only scalar)

**Scalar feature selection** (92) **à** we select one feature at a time **à** the idea is that we build a feature vector iteratively, starting from an empty feature vector and adding one feature at a time. At each step we include the feature that maximizes the function J (see formula on slide 92). In the formula the correlation can go from 0 (no correlation) to 1 (complete correlation). k is the number of features already present in the feature set. The coefficients α\_1 e α\_2 are the weights of the 2 parts of the formula; if α\_1>α\_2 we are advantaging the class separability, else the correlation.

Skip vectorial feature selection

**FUNCTIONS FOR THE CLASS SEPARABILITY CRITERIA: SCALAR MEASURES**

* **Kullback J Divergence à** see formula slide 94 **à** d=D\_ij+D\_ji (d is not a distance, so it’s not granted to be positive). We can expect that d will be 0 if the 2 pdfs are identical, ≠0 if pdfs are different, so we can use this as a separability criterion.
* **Fisher Discriminant Ratio (FDR)** **à** see formula slide 95 **à** it assumes that the distributions are gaussian.

What changes in this case is the denominator (the numerator is the same); in the first case the variance is lower and therefore the FDR will penalize the distributions with large values of the variance. 

**FINE SOUND CLASSIFICATION**