OPTIMAL SEGMENTATION TECHNIQUES FOR PIECEWISE STATIONARY SIGNALS

THÈSE N° 1993 (1999)

PRÉSENTÉE À LA SECTION DE SYSTÈMES DE COMMUNICATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES TECHNIQUES

PAR

Paolo PRANDONI

laurea in ingegneria elettronica, Università degli Studi di Padova, Italie

de nationalité italienne

acceptée sur proposition du jury:

Prof. M. Vetterli, directeur de thèse
Prof. K. Ramchandran, rapporteur
Prof. B. Rimoldi, rapporteur
Dr M. Thuillard, rapporteur
Dr J.-M. Vesin, rapporteur

Lausanne, EPFL
1999

Optimal Segmentation Techniques for Piecewise Stationary Signals

Paolo Prandoni

15 March 1999

Contents

Abstract

The concept of stationarity is central to signal processing; it indeed guarantees that the deterministic spectral properties of linear time-invariant systems are also applicable to realizations of stationary random processes. In almost all practical settings, however, the signal at hand is just a finite-size vector whose underlying generating process, if we are willing to admit one, is unknown; in this case, invoking stationarity is tantamount to stating that a single linear system (however complex) suffices to model the data effectively, be it for analysis, processing, or compression purposes. It is intuitively clear that if the complexity of the model can grow unchecked, its accuracy can increase arbitrarily (short of computational limitations); this defines a tradeoff in which, for a given data vector, a set of complexity/accuracy pairs are defined for each possible model. In general one is interested in parsimonious modeling; by identifying complexity with "rate" and model misadjustment with "distortion", the situation becomes akin to an operational rate-distortion (R/D) problem in which, for each possible "rate", the goal is to find the model yielding the minimum distortion.

In practice, however, only a finite palette of models is available, the complexity of which is limited by computational reasons; therefore, the entire data vector often proves too "nonstationary" for any single model. If the process is just slowly drifting, adaptive systems are probably the best choice; on the other hand, a wide class of signals exhibits a series of rather abrupt transition between locally regular portions (e.g. speech, images). In this case a common solution is to partition the data uniformly so that the resulting pieces are small enough to appear stationary with respect to the available models. However, if the goal is again to obtain an overall modeling which is optimal in the above R/D sense, it is necessary that the segmentation be a free variable in the modelization process; this is however not the case if a fixed-size time windowing is used. Note that now the reachable points in the R/D plane are in fact indexed not just by a model but by a segmentation/model-sequence pair; their number therefore grows exponentially with the size of the data vector.

This thesis is concerned with the development of efficient techniques to explore this R/D set and to determine its operational lower bound for specific signal processing problems. It will be shown that, under very mild hypotheses, many practical applications

dealing with nonstationary data sets can be cast as a R/D optimization problem involving segmentation, which can in turn be solved using polynomial-time dynamic programming techniques. The flexibility of the approach will be demonstrated by a very diverse set of examples: after a general overview of the various facets of the dynamic segmentation problem in Chapter 2, Chapter 3 will use the framework to determine an operational R/D bound for the approximation of piecewise polynomial function with respect to wavelet-based approximation; Chapter 4 will show its relevance to compression problems, and in particular to speech coding based on linear prediction and to arithmetic coding for binary sources; finally, in Chapter 5, an efficient data hiding scheme for PCM audio signals will be described, in which the optimal power allocation for the hidden data is determined with respect to the time-varying characteristics of the host signal.

Sommario

Il concetto di stazionarietà ricopre un ruolo fondamentale nel campo del trattamento dei segnali. Esso infatti garantisce che le caratteristiche spettrali dei sistemi lineari e tempo-invarianti progettati in un contesto deterministico siano ugualmente valide per le singole realizzazioni di un processo stocastico stazionario. In pratica, tuttavia, il segnale a disposizione è semplicemente un vettore di dimensione finita il cui processo generatore, pur ammettendone l'esistenza, rimane sconosciuto; postulare la stazionarietà del segnale è in questo caso equivalente ad ammettere che un singolo sistema lineare tempo-invariante sia sufficiente a modellizare i dati nella loro interezza, vuoi per fini di analisi, vuoi per fini di elaborazione e compressione. È chiaro, intuitivamente, che se la complessità del sistema lineare potesse crescere illimitatamente, così crescerebbe l'accuratezza della rappresentazione fornita; questa osservazione mostra una tipica situazione di compromesso in cui ad ogni possibile vettore di dati corrisponde un insieme di coppie "complessità/accuratezza" legate all'insieme di possibili sistemi lineari. Se associamo al concetto di complessità quello di quantità di dati usata nella descrizione[1] e al concetto di accuratezza quello di distorsione ritroviamo un classico problema *rate-distortion* (R/D) di carattere operazionale: per ogni ammontare della quantità di dati usata nella descrizione dobbiamo trovare il modello che, usando al massimo la quantità prescritta, fornisce una rappresentazione con minor distorsione possibile.

In pratica, tuttavia, abbiamo a disposizione solo una gamma assai ristretta di modelli, considerato che la loro complessità massima è limitata da ragioni di carattere computazionale. In questa situazione il vettore di dati nella sua interezza può dimostrarsi troppo "nonstazionario" per un modello solo. Se il processo generatore cambia lentamente, il problema può essere affrontato con successo tramite l'uso di modelli adattativi; molti segnali sono però caratterizzati da una serie di brusche transizioni che separano zone di carattere relativamente regolare: il parlato e le immagini ne sono tipici esempi. In questi casi la soluzione più comune è quella di segmentare in maniera uniforme il segnale di modo che i pezzi ottenuti siano sufficientemente piccoli da risultare stazionari

[1] Il concetto di *rate*.

rispetto alla gamma di modelli lineari a disposizione. Tuttavia, se l'obiettivo è ancora quello di ottenere una modellizzazione globale che sia ottimale nel senso R/D descritto sopra, è necessario che la segmentazione sia una variabile libera nel processo di modellizzazione, e questo non è vero nel caso si usi una suddivisione uniforme del segnale basata su una finestra temporale di dimensione fissa. Al contrario, nel caso di una segmentazione dinamica le coppie complessità/accuratezza menzionate in precedenza (che corrispondono a diversi punti sul piano R/D) sono individuate non da un singolo modello per i dati ma da una determinata segmentazione e, al contempo, da una sequenza di modelli che sarà applicata ai segmenti. In teoria, dunque, il numero di punti R/D cresce esponenzialmente con la lunghezza del vettore di dati.

Questa tesi si prefigge di analizzare e di sviluppare degli algoritmi efficienti che determinino l'insieme dei punti R/D (e il suo limite inferiore di distorsione) per specifici problemi del trattamento dei segnali. Sotto ipotesi assai blande, molte applicazioni pratiche possono infatti essere reinterpretate come un problema di ottimizzazione R/D che includa una segmentazione dinamica dei dati. A sua volta, quest'ultimo problema puó essere risolto in tempo polinomiale tramite tecniche di programmazione lineare. La flessibilità di quest'approccio sarà evidenziata da svariati esempi: dopo aver analizzato in dettaglio il problema della segmentazione dinamica nel secondo capitolo, il terzo capitolo mostrerá l'uso dello schema proposto per validare dei limiti R/D teorici concernenti l'approssimazione di funzioni polinomiali a tratti, confrontando i risultati ottenuti con quelli di uno schema di approssimazione basato sulle wavelets. Nel quarto capitolo verranno illustrati due problemi pratici legati alla compressione dei dati, ed in particolare alla compressione del parlato tramite codifica LPC e alla compressione di sorgenti binarie tramite arithmetic coding; entrambi i problemi, si mostrerà, possono offrire risultati migliori tramite l'uso di tecniche di segmentazione dinamica. Per concludere, nel quinto capitolo sarà sviluppato un sistema di mascheratura dei dati[2] per segnali musicali in codifica PCM.

[2] *Data Hiding*

Acknowledgements

No pain, no pain.
– ANONYMOUS, *workout T-shirt*

The overall readership of a Ph.D. thesis is, to put it mildly, quite limited; the technical public at large is more likely to come in contact with the associated *feuilleton* of journal and conference papers a thesis is generally borne out of and, sometimes, milked out of even postmortem. There might however be one or two epigonous students charged with the dubious honor of "continuing the present line of research"; they will probably have to read the whole manuscript and to them I instantly offer my sincerest apologies for any lack of clarity and for the boring stuff they'll have to wade through. The only other qualifying readers are of course the members of my thesis committee; to them, the above apologies are likewise extended, and complemented by my gratitude for their patience and their time (if they pulled past Chapter 1, that is). Yet, more people will cast a glance on this work than just interested or coerced readers, for it is well known that fresh doctors' homes (and bathrooms) are strewn with copies of the thesis left in full view for the benefit of visiting friends; most doctors (it is rumored) also succumb to an unconfessable urge and mail copies even to their more distant and unsuspecting acquaintances. Politeness on the readers' part demands that they at least thumb through the pages; politeness on the writer's part demands that he try to put their names in print here: that's all they'll be looking for anyway. But let's proceed with order.

This thesis sums up my research efforts as a member ot the LCAV, the audiovisual communication lab of the EPFL. The lab was (and still is) under the ironhanded rule of my advisor, *monsieur le professeur* Martin Vetterli; a man who unflinchingly snatched me away from my blissful life in California, from my sushi dinners, and from my promising career in Corporate America, and who chained me to the fifth pillar of the LCAV dungeons till the thesis was done: he shouldn't be getting many thanks here. But the fact is, his incredibly

enthusiastic approach to research and his unbelievable patience towards the innumerable self-centered little life crises of grad students such as myself really give a meaning to the word "advising" which should become a worldwide academic standard. So – uhm – well – thanks, boss. And since we're talking about LCAV, let me also thank in a stride my one-of-a-kind office mate Pina, whose patience I have tried galore (and she mine) but who, amazingly, is still on friendly terms with me; and all the rest of the gang, with whom there may have been ups and downs of course, but after all a lab is a microcosm that mirrors life in all its facets and all's well that ends well. On the other side of the ocean, my gratitude goes to Grace, Masoud and Vivek for sharing with me the initial pains of relocating to Switzerland. They later returned to California, I didn't. I'll catch you later, guys. I also owe them a lot in more academic terms, of course. Mike[3] mentioned me in his *book*, so I must reciprocate; he's been the only fellow grad student I've met who could effectively counterbalance my engineering nihilism; and his unswerving course on the road to optimality has been unquestionably inspiring. Many thanks also to Metin and Les for the rewarding time (rewarding in many ways but, alas, not monetarily) I was allowed to spend at C-Cube.

This is definitely not the place to exhume unresolved and unresolvable issues of Freudian caliber with my family, so I won't justify my refusal to follow the usual custom and thank my parents. But I have just mentioned them, haven't I, which means I *am* reaching out; now it's up to them to come forward and start apologizing for not buying me that beautiful red toy I wanted so much when I was three and a half. My sister, on the other hand, I will thank – hoping one day she'll stop chiding me for what she perceives as an immoral lifestyle. Friends are fortunately easier to deal with; Jochen (lawyer and musician), Mike[4] (historian and musician), and Nicola (engineer and musician) have provided many an hour of respite from the soulless labor of engineering research: the common denominator should be apparent; Jochen is also especially credited for his amazing herbal remedies which helped me through these harsh Swiss winters. My old high-school buddies (Berga, Calde, Ceck, Rige, plus their significant others), though all balding, kept making my trips back to Italy a surefire blast; and Monica kept reminding me why Italian women are so coveted worldwide. Thanks to Edouard and Diana for their unmatched hospitality during my trips to Berkeley and for their soul-searching facilities. And, even though I know she doesn't want to be mentioned, I really could not move on to business without a huge *merci* to Claire.

Lausanne, April 27, 1999.

[3] Goodwin
[4] Gubser

Chapter 1

Introduction

Non domandarci la formula che mondi possa aprirci
Sì qualche storta sillaba e secca come un ramo.
Codesto solo oggi possiamo dirti,
Ciò che non siamo, ciò che non vogliamo.

– E. MONTALE, *Ossi di seppia*

Never trust anybody above 30.

– Banner at Sather Gate, Berkeley

The most eventful moment in the academic journey of an engineer-to-be is without doubt the study of Taylor series; from that moment on, no natural phenomenon proves too complex to escape linearization and all difficulties can be dismissed by virtue of sufficiently small intervals. In signal processing, this philosophical frame of mind finds its parallel in the concept of stationarity and in the use of *sufficiently small* analysis windows. The story goes more or less like this.

Digital signal processing, in its basic form, is admittedly hardly more than a lengthy exercise on complex exponential functions. Complex exponentials' claim to fame is being

the eigenfunctions of linear, time invariant (LTI) difference equations; and it is true that a very wide array of interesting transformations of a signal can be expressed by means of LTI difference equations. The trinity at the core of DSP is therefore made up of: signals, i.e. linear combinations of complex exponentials; spectra, i.e. the sequences of coefficients in the above linear combinations; filters, i.e. LTI difference equations. Unfortunately, when this paradigm is applied to the real world, it is almost always impossible to fit the observed data into a fixed combination of basis functions, first and foremost because we cannot fathom all the minute details of a physical system. Enter stochastic processes.

The problem with stochastic processes, however, is that once we assume the data is random we have to gather statistics of all orders to have a complete description of what's happening: an impossible feat, clearly. Again, in the remarkable style of Alexander the Great facing the famous knot, engineers decide that second order statistics suffice and, whether by chance or by design, it so happens that if mean and variance are also shift-invariant then the entire LTI machinery developed for deterministic signals retains its applicability and meaningfulness. Signals this nice are called *stationary*; signals not this nice are usually winked at and welcomed to the flock anyway; and, as a crown result, Wold's *theorem* actually says that any stationary random process can be represented as filtered white noise. We're back in business with our LTI difference equations, apparently.

But the real world is not so easily tamed; and even if it is true that many signals might look like stationary processes, they do not do so for too long; they usually have a finite duration, maybe even longer than the average research grant, but finite nonetheless and this disrupts stationarity already. In addition to that, real physical systems exhibit drifts, jumps, and all sorts of changing patterns which cannot be suitably described by time-invariant second order statistics. Here the engineer remembers Taylor's lesson and chops the data into pieces: what looks complicated as a whole, is usually more digestible in small bits. Nonlinear things now look linear, nonstationary things, stationary; all this provided that the interval of observation is small enough. The philosophy is more or less this: what we're looking at through these little time windows are actually pieces of very well behaved processes, sort of Platonic *forms* showing through for a little while, and we can concentrate on them one at a time. But, in so doing, there's not much holistic randomness left in the end and the elegant premises leading to the legitimization of LTI processing are somehow disfranchised. Yet the algorithms work, and they work well.

In this thesis, therefore, far from trying to redress the situation, we are willing to carry it even further. Jump processes are not stationary, not Gaussian; we'll look at them simply as a collection of scalars, a single realization of a generating random process so beyond our descriptive grasp we won't even try to set up credible statistical hypotheses for it, other than it is piecewise regular. And we will show that if we have to go about chopping these signals in order to process them, then there is a subtler way to do so than just

splicing them into equal, small pieces. An *optimal* way, in fact.

Ideally, we would like to split the data so that these artificial breakpoints fall close to meaningful time instants, possibly matching the actual moments of change in the underlying generating system. Setting a breakpoint blindly (or at regular time intervals) is tantamount to placing a bet on the selected point: is this one a winner, i.e., does this correspond to a real physical change? Losing the bet might imply either severe undermodeling (breakpoint too late) or high redundancy (breakpoint too soon). An optimal segmentation strategy, needless to say, would always win the bet. But how can one always win? It's actually this simple: by knowing the future. This is indeed the price we'll have to pay for optimality, in the sense that we will have to consider the whole of the data before going on with the processing. For real-time applications this is bad news, of course; but, for those who can wait, the gain can be substantial, as we will show in what follows.

1.1 Beyond the fixed-window approach

1.1.1 The role of stationarity

A discrete-time random process $x[n]$ is called wide-sense stationary (or stationary, with abuse of language) if its first and second order moments are shift-invariant, i.e., if the mean is a constant and if the autocorrelation[1] depends only on time lag and not on absolute time index:

$$E\{x[n]\} = m_x \tag{1.1}$$

$$E\{x^*[n]x[m]\} = r_x[m-n]. \tag{1.2}$$

Similarly, processes $x[n]$ and $y[n]$ are said to be jointly stationary if they are both stationary *and* if $E\{x^*[n]y[m]\} = r_{xy}[m-n]$. In the majority of cases, the autocorrelation $r_x[n]$ (a deterministic signal, in itself) is absolutely summable, and therefore it possesses a valid Fourier transform $R_x(e^{j\omega})$; this is called the power spectrum of the stochastic process, and it represents the frequency distribution of the process's power[2]. It is easy to show that,

[1] We will always assume zero-mean processes, so that covariance and correlation coincide.

[2] Just as not to lose sight of the operational approach which we will follow throughout, it is maybe useful to point out right away how these correlation-derived quantities can be seen (and *will* be seen) mostly as an asymptotic abstraction of experimental measurements. In Chapter 4 we will indicate how the normal equations in linear estimations derive from a deterministic Least Squares problem. As far as the power spectrum is concerned, it is easy to see that, if we are given a N point realization of a stochastic process $x[n]$, we can always write out the following empirical spectral power distribution:

$$X_N^2(e^{j\omega}) = \left| \sum_{n=0}^{N-1} x[n]e^{-j\omega n} \right|^2 \tag{1.3}$$

given a stationary process $x[n]$ as the input to a LTI filter of impulse response $h[m]$, the output process $y[n]$ is itself stationary and jointly stationary with x and its autocorrelation is

$$r_y[n] = r_x[n] * g[n] \tag{1.5}$$

where $g[n]$ is the *deterministic autocorrelation* of the filter's impulse response:

$$g[n] = \sum_m h[n]h[m+n]. \tag{1.6}$$

By taking Fourier transforms we have

$$R_y(e^{j\omega}) = |H(e^{j\omega})|^2 R_x(e^{j\omega}); \tag{1.7}$$

it is therefore clear that a LTI filter designed to have certain spectral properties for deterministic signals such as a low- or high-pass characteristic, will retain those properties in the case of stochastic signals. It is precisely in this way that one can show the validity of the sampling theorem for wide-sense stationary data. Just as importantly, since any stationary stochastic process can be represented by an associated deterministic signal, i.e. the autocorrelation, all LTI processing can be conveniently designed in this context. It must be said that the relation between a process and its autocorrelation is not uniquely invertible since, roughly speaking, we are losing the phase information; yet, since most applications are concerned mainly with power levels, the loss is not dramatic.

In adaptive signal processing techniques, stationarity plays an equally fundamental role. Adaptivity can always be looked at as an estimation problem, since the adaptive system has to compute its operating parameters from one or more realizations of a stochastic process. Stationarity, in this context, ensures that a single set of paramenters is able to capture the entirety of the process. Amongst the various estimation criteria which are possible, the *linear* mean square error (LMSE) certainly deserves a special mention; here the fundamental assumption is that the parameters to be estimated are just a linear combination of the observed data. Together with the stationarity assumption for the underlying process, this leads to a convenient set of linear equation known as the *normal equations* and

---

which can in turn be expanded as:

$$X_N^2(e^{j\omega}) = \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} x[n]x[m]e^{-j\omega(n+m)}; \tag{1.4}$$

by a change of variable $k = m + n$, the above expression reveals itself clearly as the squared DFT of the empirical sample autocorrelation of the sequence. For stationary, ergodic processes, the latter converges asymptotically to the true autocorrelation, yielding back the standard definition of the power spectrum. But this also means that, as long as finite-support sequences are involved, we can always compute a direct FFT of the data lightheartedly.

to the corresponding MSE optimal solution (in expectation) for the estimated parameters; the Wiener filter [39] is derived precisely in this context. While the LMSE model is in fact a non-general approach to estimation, it is supported by the following observation: if we assume that the process $x[n]$ in Equation (1.5) is unit variance white noise, with autocorrelation $r_x[n] = \delta[n]$, then we see that the autocorrelation of the output process is completely definable by a proper choice of $h[m]$ (Wold's theorem) and, with this signal model, the estimation process for any stationary process is equivalent to determining the coefficients of a linear filter. In this case one can show that the LMSE estimator coincides with a general MSE estimator and the estimator is optimal in a Maximum a Posteriori (MAP) sense [6].

1.1.2 Piecewise stationarity

Basically all real-world signals, or at least all signals of any practical interest, depart from stationarity in some degree; this is a consequence of the fact that the physical systems which generate the measured data change their characteristics over time. Accordingly, we can divide nonstationary processes into two broad categories: drifting processes and jump processes. In the fist case, nonstationarity is due to smooth, "continuous" changes in the system's parameters; an example could be a chirp-like signal, in which there is a smooth shift in the main peak of the power spectrum. This class of signals is usually best handled by pointwise backward adaptive systems, which can gradually update the estimated parameters of the processes alongside with the drift [62]. Jump processes, on the other hand, exhibit sharp transitions between different operating modes of the generating system and this is the class we will consider in this work. These processes can be describes as *piecewise stationary* since, in between jumps, their parameters stay constant and they appear stationary.

Stationarity is in the eye of the beholder

The theoretical definition of stationarity, if followed to the letter, would declare nonstationary each and every signal we can actually handle since, formally, a finite-support signal is not stationary anymore even if it is a truncated portion of a stationary process. These difficulties are usually dismissed by means of suitable boundary conditions or by relying on the notion of cyclostationarity, where the moments become periodic functions of the lag.

Jump processes, however, lend themselves to more philosophical speculations about the nature of stationarity. Consider the simple example of a continuous-time random process defined as:

$$y(t) = \begin{cases} x_1(t) & \text{if } t < T \\ x_2(t) & \text{if } t \geq T \end{cases} \tag{1.8}$$

where $x_1(t)$ and $x_2(t)$ are two stationary processes with mean m_1 and m_2 respectively. Assume T, the switching instant, is a known deterministic quantity; then the mean of the global process is

$$E[y(t)] = \begin{cases} m_1 & \text{if } t < T \\ m_2 & \text{if } t \geq T \end{cases} \tag{1.9}$$

which clearly indicates $y(t)$ is nonstationary. On the other hand, assume we regard T as a random variable, independent of x_1 and x_2; assume, for instance, that T is uniformly distributed over an interval $[A, B]$. In this case the mean is still dependent on time but, as the interval extends to cover the entire real line, it approaches the asymptotic limit of $(m_1 + m_2)/2$, independent of t, which leads to a bona-fide stationary process (the same holds for the variance). In other words, the more knowledge we give up about the switching instant, the more stationary the process appears. Is this trade-off an advantageous one? Not if all we have at hand is just one (or a few) realizations of the process. In this case all parametric estimations must proceed from time averages, exploiting the ergodicity of each of the underlying processes. Knowing the structure of the data, namely that there is a switch, and determining its location first, improves the performance of this approach. Renouncing all knowledge about the switch might fulfill a formal stationarity hypothesis but clearly undermines the practice-friendly ergodicity assumption[3].

Although more rigourous mathematical characterizations of piecewise stationary processes can be adopted [27], for our purposes it will suffice to consider the qualitative notion of a global process in which an *unknown yet deterministic* switching pattern chooses the current output from a set of stationary and independent random processes. A very common type of signal which can be described by this paradigm is human speech, for instance; in the classic source-resonator modelization, we can recognize source switches (between voiced and unvoiced sounds) which correspond to switches between statistically different random sources; and we can recognize resonator switches (corresponding to the different articulatory postures of the vocal apparatus) which alter the spectral properties of the sources. Consider now a hypothetical experiment in which the same sentence is uttered several times in a row; the switchpoints will fall approximately at the same time instants while the random processes corresponding to the actual sounds will exhibit their typical variability across realizations, which can be exploited to infer a more accurate parametrization of the processes themselves. The operational hypothesis of a deterministic switching pattern finds its algorithmic realization in the concept of *time windowing*, in which the signal is split into smaller pieces prior to analysis.

[3]Similary, regularly switching signals (such as modulated carriers in communications) can be made cyclostationary in theoretical analyses via the introduction of an initial random delay uniformly distributed over the modulation duty cicle [39]; this however does not affect practical detection schemes, which (rightly so) first determine the synchronization delay (if any) and then assume a deterministic switching pattern.

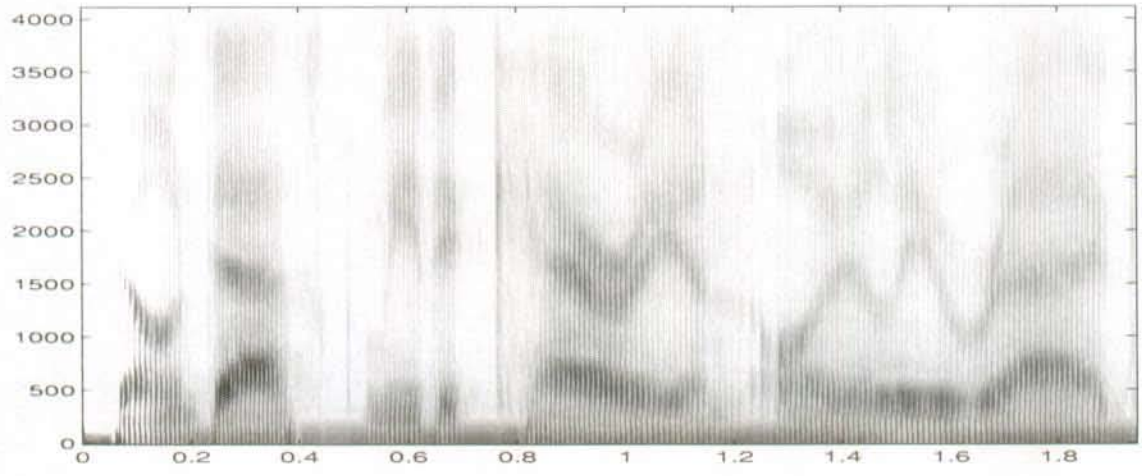


Figure 1.1: Wideband Spectrogram

Basic windowing

Jump processes are usually handled by time windowing techniques using small windows, in the hope that over the majority of intervals no jump occurs; inside these windowed portions, processing can therefore proceed as for a generic stationary signal, meaning we can access the whole body of LTI analysis tools developed in the deterministic setting. In its crudest form, windowing simply splits the signal into small segments of fixed size, the so called "fixed-window" approach, where the size of the window is small enough compared to the rate of change of the underlying physical process. This is tantamount to assuming a regularly spaced set of switching instants. We should note three problems right away: first, the notion of "small enough" is generally hardly quantifiable; second, in reducing the support of the signal to that of a small segment, we are automatically providing a limited amount of data to the processing tools, which might render unsatisfactory the results of the analysis itself; third, process switchpoints will still fall inside some of the windowed segments, and not necessarily on their boundary.

The most common tradeoffs of this fixed window approach are exemplified by a very common tool in speech analysis called the *spectrogram*. The spectrogram is a two-dimensional plot of the short-time Fourier transform (STFT) of the signal; the STFT is in turn obtained by sliding a fixed-size window over the signal and by computing a discrete Fourier transform of the corresponding segment. A first approach is that of picking the window small enough so that, in general, at most a single jump falls within a segment; the resulting spectrogram is called *wideband* and a typical plot is displayed in Figure 1.1. The rationale for the name is apparent once we recall that the DFT can be interpreted as a

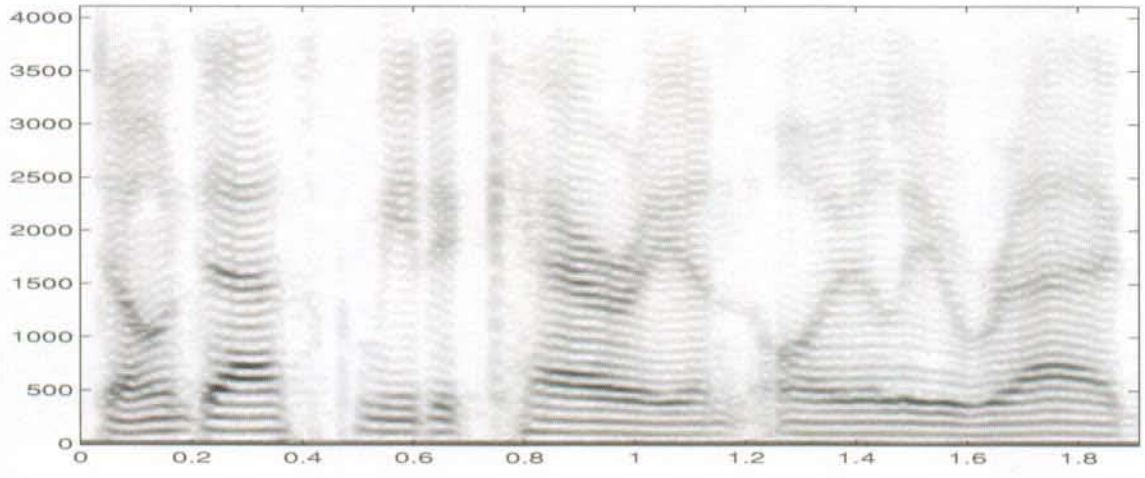


Figure 1.2: Narrowband Spectrogram

rudimentary modulated filterbank whose prototype passband characteristic is the Fourier transform of the time-domain data window [19]. A small window will necessarily originate a wide main lobe in the frequency domain, and the frequency resolution of the transform will be therefore limited, from which the label "wideband". A window small enough to characterize transients is often too short compared to the period of voiced sounds; the result is that the corresponding DFT's are not able to resolve the harmonic structure of voiced speech. The alternative is to use a longer window, which produces the *narrowband* spectrogram of Figure 1.2, but in this case the transients are not delineated clearly since the location of the jump is "diluted" over the window support. These resolution problems, it is important to point out, are not a consequence of the chosen transform (a Fourier transform rather than any other linear transform), but just of the fixed window size.

1.1.3 Dynamic segmentation

From the spectrogram example it is evident that, in order to achieve a good analysis of a piecewise stationary signal, the fundamental step is to arrive at a suitable, time-varying segmentation of the data in which the breakpoint correspond to the actual jumps and to the jumps only. However, the fundamental difficulty stems from the fact that, in general, what we have at hand is just one realization of the process (think of speech, images); the process being nonstationary, all ergodicity assumptions obviously fail and it is therefore impossible to infer a probabilistic model of the switching pattern. The only viable alternative, therefore, is to perform some sort of *operational* stationarity test on the observed realization; but the key point to note here is that, in the context of experimental data, there

is no *absolute* notion of stationarity: any such measure is entirely related to the ultimate analysis we intend to carry out and to the data models we will use. In an operational framework, therefore, the word "stationarity" is in itself a misnomer; better would it be to use the term "regularity", for instance, or "model matching". However, given that the term has a well-understood connotation even in an empiric context, we will retain it for clarity.

Best adaptive bases: a classic example

Prime examples of dynamic segmentation depending both on the analysis tools and on the data realization are to be found in the context of data compression. Lossy compression by transform coding is based on the *energy compaction* property of some classes of linear transforms. This property ensures that, in the transform domain, the characteristics of the original signal are adequately represented by a small subset of basis vectors; by appropriately thresholding the transform coefficients, an arbitrary compression factor can be achieved. Linear transforms, however, assume a stationary data set; in response to this difficulty, Coifman and Wickerhauser introduced the concept of best adaptive basis [9]. Their fundamental idea is the definition of an additive cost functional $\mathcal{M}(\cdot)$ which measures the effectiveness of a local basis. Depending on the properties of the basis, a recursive algorithm splits either the frequency or the time axis into diadic subintervals, for an optimal fit of a series of local basis to a finite data vector; wavelet packets are used in the frequency domain and local trigonometric bases in the time domain. The setting has also been extended to more general time/frequency segmentations (and more general basis functions), leading to arbitrary tilings of the T/F plane [20]. Yet this approach would be of limited practical value if it didn't take explicitely into account the cost of representing the data in the new basis. This cost appears clearly after explicit quantization of the coefficients and affects the faithfulness of the representation via the associate distortion. This further extension of the best basis framework to the case of transform coding for lossy compression was introduced by Ramchandran and Vetterli in [45]. As stated previously, in lossy compression we are often interested in quantizing the data in the transform domain; the goal is to find the best basis while also trying to minimize the corresponding distortion. In the quantized domain, we can associate a precise cost R in terms of bit rate to each basis, and a corresponding distortion D; the main idea in [46] is to look at the cost measure $\mathcal{M}(\cdot)$ introduced above as an additive cost measure of Lagrangian form $R + \lambda D$, which induces a shift from the simple set of local transforms to the more meaningful set of *quantized* representations in a local basis. The goal is once again to pick the sequence of local expansions which minimizes $\mathcal{M}(\cdot)$, with the proviso that the overall cost of the representation must fall below a certain given rate budget.

Best basis algorithms are a typical instance in which we do away with stationarity assumptions or stationarity tests but simply look for the *jointly* optimal time segmen-

tation and set of transforms for a particular data set. It is clear that from a theoretical point of view one could define a class of stochastic processes all of which lead to the same best basis; from a practical point of view, however, we are willing to pay a computational price in order not to formulate assumptions on the data and let the algorithm implicitly define the class by exhaustively exploring the space of all bases.

More general coding models

The previous framework can be extended to include more general data models. This comprises the set of local transforms introduced above, of course, and reaches out to a wealth of linear and nonlinear processing schemes, both adaptive and nonadaptive. In particular, classic schemes such as Linear Predictive Coding (LPC) [44], sinusoidal modeling [41], local polynomial modeling [43], and arithmetic coding [42] have all been shown to benefit from a dynamic segmentation framework; the following chapters will indeeed illustrate the point in detail. Again, the fundamental concept is the definition of a cost function which measures the descriptive performance of a given algorithm together with its coding cost over a given data segment, and which implicitely constitutes an operational stationarity test; now, however, we will not expect this function to be strictly additive. For arbitrary time segmentations, this introduces some extra complexity and part of this work is to illustrate how this can be tackled in an algorithmic way.

We have already said that a misplacement of the breakpoints in a segmentation can lead to either redundancy in the representation or to undermodeling. Perhaps one of the most interesting facets of dynamic segmentation is that the placement of breakpoints (and the corresponding modelization) can be seen as a function of the amount of resources we choose to spend. This defines a unified framework which bridges modeling, lossy compression and lossless compression. In pure modeling problems we are interested in minimizing some distortion measure, regardless of the cost of our representation. In lossless compression we are interested in minimizing the cost instead, while using non-distorting coding models. The case of lossy compression lies in between. By suitably segmenting the data, all three cases are just outcomes of the very same algorithmic setup.

As a final note, it is important to remark that, by allowing for general basis or coding models, the necessity arises to efficiently represent the information relative to the segmentation structure; this is particularly evident in applications such as compression. While a dynamic segmentation of the data together with an optimal sequence of compression algorithms usually yields a substantial bitrate reduction with respect to a fixed scheme, the cost of the side information needed to identify the segmentation might largely undo this gain. Obviously, side information costs must be integral part of the dynamic segmentation process; therefore, efficient ways to tag this information to the data have to be taken into account.

1.2 Thesis outline

The rest of the thesis is organized as follows: Chapter 2 will elaborate on the concept of optimal time segmentation and optimal sequence of data models in great detail from an abstract point of view. Once the algorithmic tools are in place, Chapter 3 will introduce the first example of optimal data modeling, which involves piecewise polynomial approximations; the corresponding operational results will be compared against two theoretical bounds on piecewise polynomial approximation derived in the same chapter, with respect to an oracle-based coding scheme and to a standard, nonlinear approximation wavelet coding scheme. Chapter 4 will apply the optimal segmentation strategy to two different instances of practical lossy compression schemes: LPC speech coding and Arithmetic Coding for binary streams. Finally, in Chapter 5, the framework will be carried into the realm of data communication, here in the form of Data Hiding; user data are to be embedded into a host signal, an audio signal in this case, and optimal segmentation techniques in conjunction with a perceptually-based cost function will allow us to maximize the overall throughput.

Chapter 2

Signal Segmentation and Resource Allocation

4.27: Bezüglich des Bestehens und Nichtbestehens von n Sachverhalten gibt es $K_n = \sum_{\nu=0}^{n} \binom{n}{\nu}$ Möglichtkeiten. Es können alle Kombinationen des Sachverhalte bestehen, die andern nicht bestehen.

– L. WITTGENSTEIN, *Tractatus Logico-Philosophicus*

The basic ingredients of an optimal segmentation problem are a finite data vector, a family of data models, and a cost function which measures the *global* goodness-of-fit of the modelization. The modelization, in itself, is characterized *jointly* by a segmentation of the data and by the sequence of data models which are applied to the segments; achieving optimality corresponds to determining the modelization (in the joint segmentation/model-sequence sense just described) which minimizes the given cost function.

In order to ground the discussion in a concrete example, consider the case of a piecewise regular function composed of a series of contiguous polynomial pieces. Assume the data vector is a N-point sampled version of such a function over a given interval; a typical example is shown in Figure 2.1. The goal is to describe these data using local polynomial functions up to a maximum degree (the family of data models) minimizing the global mean squared error (the cost function) of the representation. At first it might appear that in this case the localization of the breakpoints in the data (the segmentation)

and the fitting of a local polynomial to each piece (the modelization) could be carried out as separate, subsequent steps. However, what if some polynomial pieces have a larger degree than those available in the given family of models? What if the data are corrupted by noise? It is intuitively clear that, unless segmentation and allocation are performed jointly, there can be no guarantee of true optimality with respect to the cost function.

A related question, and one which is inextricably tied to the practical, algorithmic nature of the topic, is how to encode the modelization while preserving optimality; since the "cost" of a polynomial piece (in terms of storage, for instance) increases with its degree, in our example the question would translate to: how much do we want to spend for each segment, and how can this price affect the segmentation? Indeed, more often than not, the optimal segmentation problem involves a fourth ingredient, which is an upper limit on the amount of resources which can be spent on the modelization. This additional requirement transforms the problem into a constrained optimization and it is easy to see that it is this constraint which really makes the segmentation problem an interesting one; in its absence, for instance, the N-point piecewise polynomial vector described above can always be modeled with no error by a sequence of N polynomial "pieces" of zero degree over N "segments" one data sample wide. Setting a limit on the resources or, better, using the limit to explore a set of optimal tradeoff points for different amounts of resources, automatically eliminates such trivial solutions and opens the way to parsimonious modeling approaches; it now becomes even more apparent how the segmentation must be determined jointly with the model fitting.

The fundamental premise to the optimal segmentation algorithms we will introduce in the following is that they can all be regarded as an extension of the classic constrained resource allocation problem; such a problem arises each time a limited amount of resources must be suitably partitioned across several recipients in order to minimize (or maximize) a global cost function. The best-known such case in signal processing is probably unequal bit allocation for subband coding [61], for instance, but widely diverse examples abound in disciplines such as operation research. In the simpler case when the allocation can be performed independently amongst the different recipients, the solution to the problem obeys to a simple intuitive rule: the optimal distribution of resources is that for which the *differential* decrement (or increment) of the cost function is the same for all recipients. In other words, we favor those recipients which can make the most out of the resources they are given, not those which are most in need[1]. In the jointly optimal allocation/segmentation case, the recipients are the segments themselves and the allocation process still follows the aforementioned guideline; but, since the segmentation is a free variable in the allocation process, some complications arise: dependence amongst allocation choices is the primary difficulty and much of this work is devoted to finding efficient formulations for this case.

[1] A heartless, meritocratic policy one could argue, yet optimal in the global scheme of things.

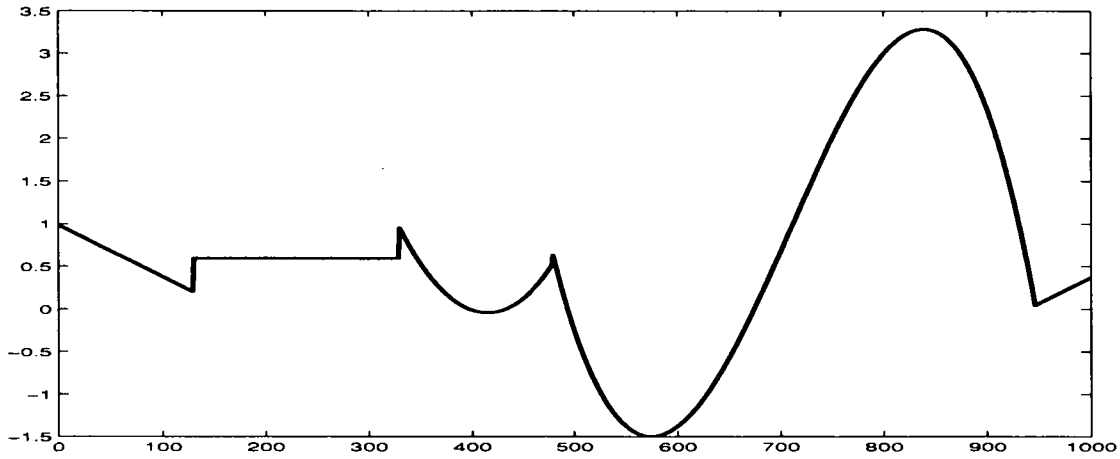


Figure 2.1: An example of piecewise polynomial function.

The rest of the chapter is organized as follows. We will first briefly compare the philosophy behind the operational segmentation/allocation framework to the related (but eminently theoretical) subject of rate-distortion theory. We will then proceed to analyze three cases of optimal segmentation, for separable, partially separable and nonseparable cost functionals and describe algorithmic solutions based on a Lagrangian minimization. Data encoded according to an arbitrary segmentation require that side information about the segmentation itself be part of the overall description; therefore, we will then tackle the issue of efficiently encoding this side information by means of "colored bits". Finally, in the Appendix, we will consider alternative algorithmic implementations of the segmentation algorithm for the three cases above; these, based on dynamic programming, can overcome some of the limitations of the Lagrangian approach albeit at a higher computational cost.

2.1 Optimal allocations and rate-distortion theory

In signal processing, most allocation (and segmentation) problems appear in the context of data compression and/or data transmission; without much loss of generality, we can therefore rephrase the general setup in more familiar terms by identifying the values provided by the cost function as a measure of distortion and by representing the amount of available resources by its descriptive cost in bits. With these associations, it is natural to draw a parallel between the optimal allocation problem and the information theoretical concept of rate-distortion (R/D). Classic R/D theory is concerned with the approximation of a (possibly multidimensional) random source with respect to a given *fidelity criterion*, which is an

exhaustively-defined distortion measure over the cartesian product of the source alphabet and a set of reproduction symbols; it answers the question of how well this can be done via an achievable lower bound on the number of bits necessary to achieve a certain reproduction fidelity. Similarly, in our allocation/segmentation framework, we are concerned with a rate-constrained representation of the input data by means of a pre-defined set of reproduction "symbols" which are nothing but a data segment and its associated data model.

A simplified setup of the theoretical framework in the discrete case is the following [4]. We have an input source with independent, identically distributed (iid) symbol probabilities $p(a)$ and alphabet $\mathcal{A}$ and an output alphabet $\mathcal{B}$; we also have a fidelity criterion (i.e. an *additive* per-letter measure) which is basically an $M \times N$ matrix: entry $d(a,b)$ is the distortion we incur by representing input symbol a by output symbol b, for any a and b. These are the fixed parameters of the problem, known to *both* encoder and decoder. We then consider a random transformation from the input to the output alphabet, in the form of a conditional probability distribution $q(b|a)$. Auxiliary quantities induced by the conditional probability are the average distortion

$$D_q = \sum_a \sum_b p(a,b)d(a,b) \tag{2.1}$$

and the mutual information

$$I_q(b;a) = \sum a \sum_b p(a,b) \log\left(\frac{p(a,b)}{p(a)p(b)}\right) \tag{2.2}$$

where the marginals and joint probability distributions are obtained from $q(b|a)$ in the usual way. Shannon [53] showed that the *achievable* lower bound for the distortion-rate function is in this case:

$$D(R) = \min_{q|I_q(b;a)\leq R} D_q \tag{2.3}$$

in the sense that, for a given rate R, we cannot hope to encode the source with a distortion less than $D(R)$.

Let's now have a look at the situation in practice. We are looking for a deterministic transformation of the input data which, in itself, is a sequence of local transformations of the segmented input chosen from a collection of K possible coding models. In other words, the input is just a finite size vector $\boldsymbol{a}$ in which the single elements are the data segments, while the output symbols are a vector $\boldsymbol{b}$ of "codewords" identifying the encoded segments. We can identify the composite transformation by $Q(\cdot)$, so that $\boldsymbol{b} = Q(\boldsymbol{a})$; the transfomation can be inverted and an additive distortion measure is defined by

$$D_Q(\boldsymbol{a},\boldsymbol{b}) = \sum_j d(a_j,b_j) = \sum_j \|a_j - \hat{a}_j\| \tag{2.4}$$

with $\hat{\boldsymbol{a}} = Q^{-1}(\boldsymbol{b})$ and $\|\cdot\|$ a suitable error norm. Denote by $H(\boldsymbol{b})$ the cost in bits for completely identifying the sequence of output symbols. The overall goal is to find the composite tranformation Q which minimizes the global distortion while requiring less than R bits to encode the output:

$$D(R) = \min_{Q|H(Q(\boldsymbol{a}))\leq R} D_Q; \tag{2.5}$$

there is clearly a formal analogy with (2.3), even though the setup is completely different. Yet, this is one of the reasons why most of the techniques we will introduce in the following have been addressed in the literature as operationally "R/D optimal".

In the above expression, the operational rate $H(\boldsymbol{b})$ is dependent on the way the output symbols are coded, and is theoretically lower bounded by the entropy of the output sequence $\boldsymbol{b}$. However, in a truly algorithmical setup, we don't have an input source proper but a finite, size-N data realization with no better statistical description than explicit counting. We could still try to do our best by considering the set of all possible size-N vectors of output symbols b_1^N and minimize (2.5) using the empirical output entropies relative to the histogram-based distribution induced by each vector,

$$D(R) = \min_{b_1^N|NH(b_1^N)\leq R} \sum_{j=1}^{N} d(a_j, b_j); \tag{2.6}$$

the implicit mapping between the input vector and the minimizing $\boldsymbol{b}$ identifies the coding transformation $Q(\cdot)$. Yet this would be exponentially complex, since there are N^K possible output vectors for each input. So we yield even more and we make the expression for the rate additive too, which is tantamount to saying that we will code the output sequence according to a pre-defined statistical distribution $P(\cdot)$ for the K coding models, and we will pay the Kullback-Leibler price for all $\boldsymbol{b}$'s which do not conform:

$$D(R) = \min_{b_1^N|\sum_{n=1}^{N}\log_2 1/P(b_n)\leq R} \sum_{j=1}^{N} d(a_j, b_j). \tag{2.7}$$

This is the fundamental equation for all the problems we will treat in the following; we will refer to this simplified R/D scenario as to an *operational rate-distortion framework* and in the rest of the chapter we will illustrate some efficient algorithmic techniques to solve the problem in (2.7). Although most statistical assumptions on the data have been dropped (which moves us further and further from the Shannon's bound), any prior knowledge on the input is obviously taken into account from an engineering point of view. Computational requirements, for instance, call for economy in the number of available models and in their inherent complexity: indeed, little would be gained by allowing for, say, sinusoidal

models in the case of piecewise polynomial data; yet the time spent on the search for the optimum output sequence would dramatically increase. Similarly, the implicit distribution for b will be crafted to have the allocation process do what we think is "reasonable".

2.2 Segmentation and allocation techniques

2.2.1 Notation and building blocks

Here and in the following the data will be represented by a N-element vector x_1^N; the elements themselves are best viewed as sets of data samples: a single scalar value, disjoint segments of a one-dimensional discrete time signal, or 8×8 image blocks, for instance; we will see later that the choice of what is a "data point" determines the underlying granularity of the segmentation/allocation process, and is in itself a useful parameter controlling the end results. We will assume we have a family of K data models, to each of which a distortion function and a rate function pair is associated; both rate and distortion functions are dependent on the set of points the model is applied to. We can right away distinguish between two main classes of models, which will give rise to two different allocation problems:

- *Separable* models, for which rate and distortion are additive over the single data points (a family of scalar quantizers, for example);
- *Nonseparable* models, for which rate and distortion are nonadditive functions, in the sense that they depend on the range of (consecutive) data points the same model is applied to (a family of transform coders, for instance).

Both types of models can be adaptive or non adaptive, where adaptive means that the parameters used to describe a model instance are derived from the data and must be explicitely encoded along with the data and the model index; polynomial models, for instance, are obviously adaptive while a scalar quantizer is not.

2.2.2 Optimal resource allocation, independent case

Let us make a little step backward and, for the time being, forget about the segmentation issue to concentrate on the allocation problem only. For our N-point data vector and our family of K data models the goal is to assign a model to each point in order to minimize the overall distortion under a given rate constraint; assume both rate and distortion additive over the N data points, which restricts the allowed models to the separable class.

Theory: the reverse waterfilling approach

The resource allocation problem can be tackled from within two very different frameworks. In a probabilistic scenario we would assume that the data are the realization of a N-dimensional random process for which we possess a statistical description, and that the distortion for each of the N data sources can be expressed analytically as a function $d(r)$ where the rate, approximated by a real-valued quantity, indexes the model. A typical case is that of quantization, where a family of quantizers can easily be ordered according to a sufficiently dense set of rates; another example could be the truncation of local series expansions, as in block-transform coding schemes like JPEG [46]. In all cases the goal is to arrive at a global allocation scheme which is optimal in expectation but note that, somehow, this is still an operational R/D setting since the overall rate is considered additive. In many practical cases, such a rate/distortion curve is a cup convex function, and the optimal resource distribution can be efficiently determined using reverse water-pouring techniques or variations thereof [17]; the optimal distribution must satisfy the following constant slope conditions and rate constraint:

$$\begin{cases} \dfrac{\partial D}{\partial r_n} = \text{constant for } n = 1, \ldots, N \\ \\ \sum r_n = R_0 \end{cases} \tag{2.8}$$

where $D = \sum_{n=1}^{N} d(r_n)$ is the sum of the N independent R/D functions for the sources and R_0 is the overall available rate. The resulting allocation scheme can be considered "hard wired", since it is designed only once for an entire *class* of data sources and it is crystalized as an integral part of the overall data processing algorithm.

Practice: the operational approach

The second approach to resource allocation, which we will follow throughout, does away with statistical descriptions and deals exclusively with the particular data realization to be processed. While the set of K models has clearly been designed exploiting some form of prior knowledge about the data sources, in this case we no longer rely on asymptotic rate-distortion curves but rather on a discrete set of *achievable* R/D pairs. In other words, the analytic curves of the previous approach are replaced by data-dependent, discrete sets of explicitly computed operating points for all meaningful combinations of model and data point; we are thus back to Equation (2.7). The discretization of the water-pouring approach for this operational case leads to a special case of Lagrangian optimization which is amenable to very efficient algorithmic implementations; the main reference in this case is the paper by Shoham and Gersho [55], to which we refer for all formal proofs.

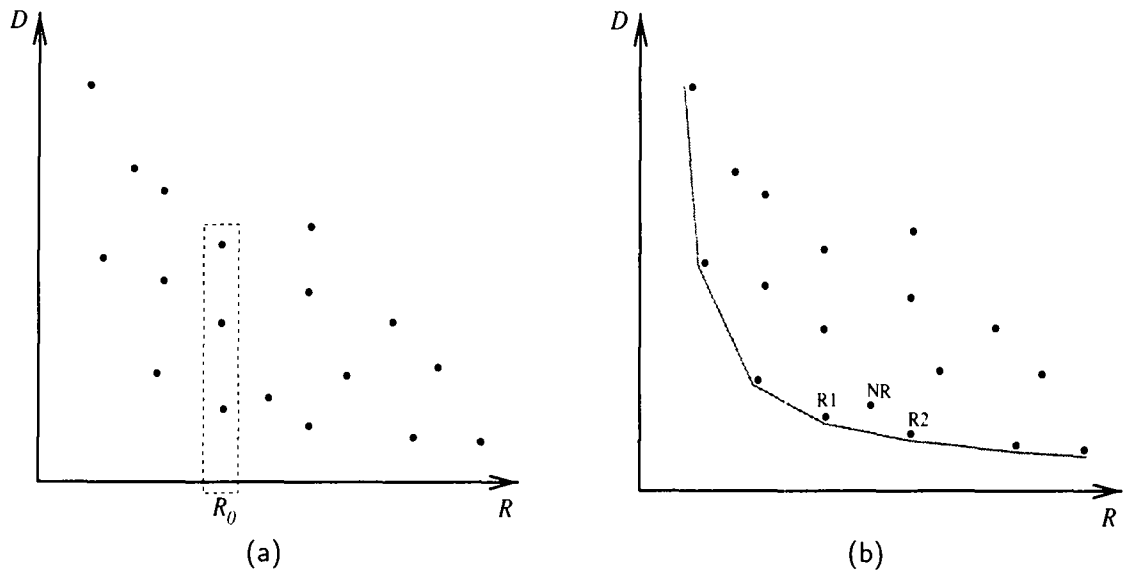


Figure 2.2: (a) Rate-Distortion pairs; (b) convex hull.

First of all, consider an arbitrary allocation $\boldsymbol{w}$ consisting in a N-element vector of model indices; rate and distortion values for $\boldsymbol{w}$ can be computed by applying the model indexed by the n-th element of $\boldsymbol{w}$ to the n-th data point x_n for $n = 1$ to N and summing up the results, since by hypothesis both rate and distortion are additive and independent; call these values $R(\boldsymbol{w})$ and $D(\boldsymbol{w})$ respectively. Given that there are K data models, the set of all meaningful allocations W has cardinality $|W| \leq K^N$ and to each $\boldsymbol{w} \in W$ is associated a R/D pair which we can represent by a point in a distortion-rate plane as in Figure 2.2-(a). The optimal allocation problem can be now formulated as the search for:

$$\begin{cases} \boldsymbol{w}^* = \arg \min_{\boldsymbol{w} \in W} \{D(\boldsymbol{w})\} \\ \\ R(\boldsymbol{w}^*) \approx R_0 \end{cases} \tag{2.9}$$

where the rate constraint is "approximate" since the ideal target rate might not be exactly amongst the discrete set of reachable operational rates. Without further assumptions, the solution of (2.9) involves an unstructured exhaustive search over K^N R/D points, first selecting candidate rates (dashed box in Fig. 2.2-(a)) and then selecting the one with minimum distortion. We can however transform the process in a much more structured (and therefore efficient) search over a linear parameter if we are willing to restrict the set of candidate points to just the convex hull of the entire R/D set, highlighted in gray in Fig-

ure 2.2-(b). The key to the efficient formulation lies in restating (2.9) as an unconstrained problem via a Lagrange multiplier λ; it can indeed be shown that if $\boldsymbol{w}_0$ is a solution to

$$\min_{\boldsymbol{w}\in W}\{\, D(\boldsymbol{w}) + \lambda R(\boldsymbol{w})\,\} \tag{2.10}$$

then $\boldsymbol{w}_0$ defines a point *on the convex hull* which solves the associated problem:

$$\begin{cases} \min\limits_{\boldsymbol{w}\in W}\{\, D(\boldsymbol{w})\,\} \\ \\ R(\boldsymbol{w}) \leq R(\boldsymbol{w}_0) \end{cases} \tag{2.11}$$

It may help the intuition to show why the set U of (R, D) pairs solutions to (2.10) indeed defines a convex line on the R/D plane. Given any two solutions (R_i, D_i) and (R_j, D_j), $R_i > R_j$, the line in the R/D plane connecting them has an (absolute) slope $\gamma = (D_j - D_i)/(R_i - R_j)$. Convexity requires that all solutions (R, D) such that $R_i \leq R \leq R_j$ lie below this line. Suppose this was not the case for a solution $(R, D) \in U$ in terms of slopes connecting (R, D) to (R_i, D_i) and to (R_j, D_j) this implies (see Figure 2.3):

$$\frac{D_j - D}{R - R_j} < \gamma < \frac{D - D_i}{R_i - R}. \tag{2.12}$$

The (R, D) pair is by hypothesis a solution to (2.10) for a given λ; however, if $\lambda < \gamma$, (2.12) implies $D_i + \lambda R_i < D + \lambda R$ and we have a contradiction; otherwise, if $\lambda \geq \gamma$, we have again $D_j + \lambda R_j \leq D + \lambda R$; therefore $(R, D) \notin U$. This holds for all elements of U and, for a dense solution set, it provides an intuitive meaning to the value of λ in (2.10) as the derivative of the convex hull at the relative solution point. It should be noted at this point that the restriction to the convex hull never affects optimality, but it might slightly reduce the density of achievable rates. Indeed, consider the point labeled 'NR' in Figure 2.2-(b); the point is optimal with respect to its associated rate, since no other point with the same rate lies below it, yet it would not be reachable by the Lagrangian optimization since it doesn't lie on the convex hull. Instead, the closest Lagrangian soulution would either be point 'R1' or 'R2', depending on λ. Usually, for sufficiently dense sets of coding models, this restriction is of no practical consequences; in Appendix 2.A, however, we will describe an algorithm based on dynamic programming which circumvents this problem.

With this unconstrained formulation the search strategy therefore involves finding the value of λ for which the solution $\boldsymbol{w}$ to (2.10) satisfies $R(\boldsymbol{w}) \approx R_0$. So far, however, little seems to have been gained in terms of efficiency. The breakthrough comes from two separate observations; first, since rate and distortion are additive, non negative and inde-

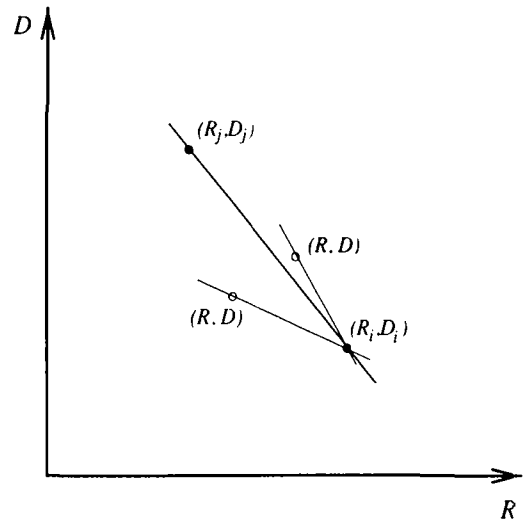


Figure 2.3: Convexity of the hull.

pendent, we can rewrite (2.10) as:

$$\begin{aligned} \min_{\boldsymbol{w}\in W}\{\, D(\boldsymbol{w}) + \lambda R(\boldsymbol{w})\,\} &= \min_{\boldsymbol{w}\in W}\{\sum_{n=1}^{N}\{d(n;w_n) + \lambda r(n;w_n)\}\,\} \\ &= \sum_{n=1}^{N}\min_{k=1,\ldots,K}\{\, d(n;k) + \lambda r(n;k)\,\} \end{aligned} \tag{2.13}$$

where $d(n;k)$ and $r(n;k)$ are the distortion and the rate associated to using model k over x_n. Since the terms in the sum are independent, the total number of comparisons needed to solve (2.10) is now on the order of KN for a given value of λ. Equation (2.10) represents in fact a discretized version of the constant slope condition in (2.8), where all points operate at the same tradeoff factor λ.

The second fundamental observation is that there is a monotonic relationship between λ and the total rate (for a proof, see again [55]). This means that the desired value can be found using a fast search, using for instance the bisection method [45]; although it is hard to give a precise estimate of the number of iterations needed to fulfill the given rate constraint, dependent as it is on the initial value for λ, it is usually remarked that the time spent on the iteration is rather small. A priori information on the characteristics of the data is usually sufficient to formulate an educated guess for the initial value which minimizes the number of refinement steps. This monotonic relationship between the rate and the Lagrange parameter is a fundamental asset of the unconstrained formulation; it

gives λ the role of a user definable "knob" with which the problems of data compression ($\lambda \to +\infty$) and data modeling ($\lambda \to 0$) are seamlessly bridged.

2.2.3 Optimal resource allocation, partially independent case

From allocation to segmentation: side information

In the operational optimization framework for additive cost functionals just described a fundamental piece is still missing: since the allocation is strictly dependent on the particular data vector, its description becomes an integral part of the overall modelization, and must be taken into account. In fact, the usual purpose of an optimal allocation is data compression; it is obvious that, by allowing for an arbitrary sequence of encoding schemes, we must provide an adequate amount of information to the decoder, on top of the compressed data, if we want to invert the process. This description, however, exacts a non-negligible toll on the total available rate in the form of side information; to preserve the optimality of the solution, this extra cost must be explicitely included in the allocation process. The question now is: what is the actual cost of encoding the allocation structure? Consider a simple example: assume that the data are stored in an 8-point vector and that the family of models is comprises 4 different coding schemes, labeled A, B, C and D; let the final allocation be A, A, A, A, A, C, C, C: we can encode this (or any other) sequence in a straightforward manner using $8 \times 2 = 16$ bits. *If the data are piecewise stationary*, however, neighboring data points will have on average a higher likelihood of being assigned the same data model and in this case some form of run-length coding is decidedly more effective. Indeed, the previous allocation can be encoded with $2 \times (3 + 2) = 10$ bits, where we use $\log_2 8 = 3$ bits to encode a run and 2 bits for the model. Run-length coding implicitely segments the data into blocks of points for which the allocation is constant; we can qualify this observation as follows:

Proposition 1 *For piecewise stationary signals, if side information is taken into account efficiently, each modelization subsumes a segmentation even in the case of an additive cost functional.*

This encoding scheme has also another important consequence: it creates dependence between data points with respect to the allocation. Indeed, the model choice for the n-th data element has now different rate requirements depending on the choice for the previous element (i.e. whether we need to start a new encoding run or extend the previous one). In this case, equation (2.13) does not hold anymore and the allocation algorithm will have to be modified accordingly. We can therefore state the following:

Proposition 2 *Side information introduces dependence in the allocation process.*

Extending the lagrangian framework

This one-step-back dependence in the cost functional is after all rather simple and the independent allocation framework can be easily extended to the present case with some extra algorithmic complexity but at no major increase in the computational requirements. In fact, the distortion values are still independent and we can integrate the cost of side information by introducing a new function for the rate $r_c(n;k,h)$ defined as

$$r_c(n;k,h) = \begin{cases} r(n;k) & \text{if } k = h \\ r(n;k) + c(h,k) & \text{if } k \neq h \end{cases} \tag{2.14}$$

where $c(h,k)$ is the cost (in bits) of signaling a transition from model h to model k; this information will comprise a model index and a measure of the number of points the model is applied to (more details about the values for $c(\cdot,\cdot)$ will be discussed in section 2.3.2). With this, the search for the optimal cost functional in Equation (2.13) becomes

$$\begin{aligned} J^*(\lambda) &= \min_{\boldsymbol{w}\in W} \{\, D(\boldsymbol{w}) + \lambda R(\boldsymbol{w}) \,\} \\ &= \min_{\boldsymbol{w}\in W} \{ \sum_{n=1}^{N} d(n;w_n) + \lambda r_c(n;w_n,w_{n-1}) \,\} \end{aligned} \tag{2.15}$$

with the convention that w_0 is set to some agreed-upon initial model. The minimization in the above expression is no longer separable; however, suppose we knew the value w_i^* of i-th element in the optimal allocation $\boldsymbol{w}^*$; since the dependence amongst terms extends only one step backward, we can write:

$$\begin{aligned} J^*(\lambda) &= \min_{w_1^{i-1}} \{ \sum_{n=1}^{i-1} [d(n;w_n) + \lambda r_c(n;w_n,w_{n-1})] + \lambda c(w_i^*, w_{i-1}) \} + \\ &+ d(i;w_i^*) + \lambda r(i;w_i^*) + \\ &+ \min_{w_{i+1}^N} \{ \sum_{n=i+1}^{N} d(n;w_n) + \lambda r_c(n;w_n,w_{n-1}) \}_{w_i = w_i^*}. \end{aligned} \tag{2.16}$$

The three terms on the right state that the global optimum is composed of the partial allocation whose Lagrangian cost, *including the cost of a potential transition to* w_i^*, is minimum up to $i-1$, plus the Lagrangian cost for the point we assume to know, plus the cost for the remaining points, which is independent of the previous ones. This holds for any value of i and therefore, at each step, the optimal allocation can be built incrementally by formulating an optimality hypothesis over each one of the K models. Equation (2.16) is a particular case of the optimality principle in dynamic programming [3] and suggests an efficient way to implement the allocation process as a trellis search using the Viterbi algorithm [15]. The

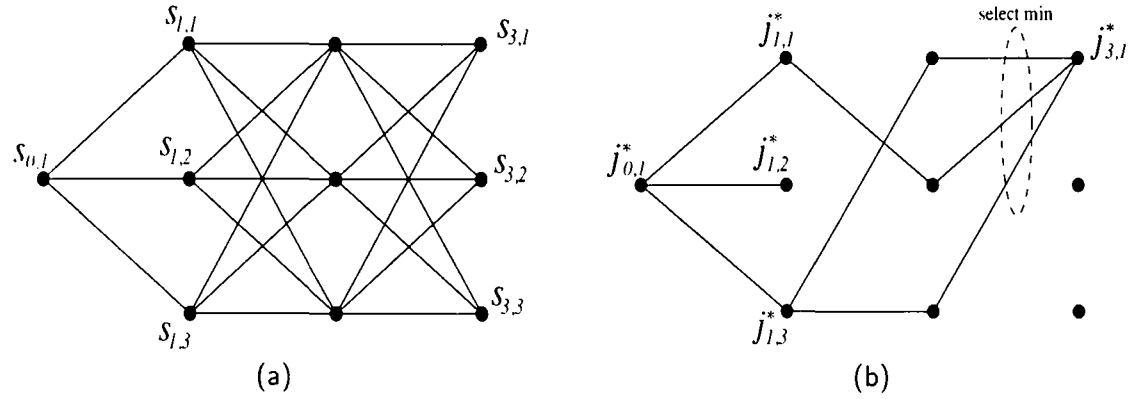


Figure 2.4: Trellis for the partially independent case: (a) trellis structure; (b) path population.

trellis formulation has the advantage of storting all intermediate rate and distortion values in a convenient structure, making an iteration over the Lagrange parameter much quicker, and is a popular algorithmic strategy in similar optimization problems [36, 56]. Note that the computational requirements are still linear in N, just as the storage requirements.

Algorithm 2.1: Trellis algorithm for the partially independent case

The minimization in (2.16) can be deployed on a K-state, $(N+1)$-stage trellis as in Figure 2.4-(a); a state will be denoted by $s_{n,k}$, where the first subscript identifies the step index (from 1 to N) and the second subscript a model index (from 1 to K); the same notation is used for the path metrics associated to the states, $j_{n,k}$.

Step 1: Initialization

1 Define (by convention) an initial state $s_{0,1}$ and set $j_{0,1} = 0$.

2 Create the trellis: for $1 \leq n \leq N$
create a state set S_n containing K states $s_{n,k}$ each, $1 \leq k \leq K$.

3 For each $s_{n,k} \in S_n$, $1 \leq n \leq N$
compute the distortion and rate values $d(n;k)$, $r(n;k)$ and associate these values to the state.

Step 2: Trellis path population

4 Select a value for λ; let $j^*_{0,1} = 0$.

5 For $1 \leq n \leq N$
For $1 \leq k \leq K$
Determine the minimum cumulative Lagrangian costs:

$$j^*_{n,k} = \min_{1 \leq h \leq K} \{j^*_{n-1,h} + d(n;k) + \lambda r_c(n;k,h)\}$$

using the values stored in the trellis[2].

Assume the minimum is for $h = h^*$: connect $s_{n,k}$ to s_{n-1,h^*} and associate $j^*_{n,k}$ to $s_{n,k}$ (see Figure 2.4-(b)).

Step 3: Backtracking

6 Select the state s_{N,k^*} with the minimum value for $j^*_{N,k}$; from this, obtain the corresponding overall rate and distortion values $r^*(\lambda)$ and $d^*(\lambda)$.

7 If the rate constraint R_0 is met
Then follow the paths backward in the trellis from s_{N,k^*} to $s_{1,0}$; the sequence of traversed states yields (backwards) the optimal allocation. Stop.

Else proceed to the iteration over λ.

Step 4: Iteration over λ

In some cases, especially when the rate constraint is specified to within a tolerance interval, an educated guess for λ can avoid the need for an explicit search. In most cases, however, the value for λ must be determined iteratively until the rate constraint R_0 is met as closely as possible. The search algorithm exploits the convexity of the solution set and proceeds as follows [45]:

8 First determine $\lambda_{\min}$ and $\lambda_{\max}$ so that $r^*(\lambda_{\max}) \leq R_0 \leq r^*(\lambda_{\min})$ (see below);

9 Choose a starting value for λ between $\lambda_{\min}$ and $\lambda_{\max}$.

[2]For $n = 1$, the only possible value for h is 1, obviously

10 Run the path population and backtracking steps (Steps 2 and 3 above).

11 If $R_0 = r^*(\lambda)$
Then the optimum is found. Stop.
Else
If $R_0 < r^*(\lambda)$
Then $\lambda_{\max} \leftarrow \lambda$
Else $\lambda_{\min} \leftarrow \lambda$;
Determine the new value as $\lambda = (d^*(\lambda_{\min}) - d^*(\lambda_{\max}))/(r^*(\lambda_{\max}) - r^*(\lambda_{\min}))$ and goto line 10.

Given the monotonic relationship between λ and $r^*(\lambda)$, an obvious choice for the initial minimum value is $\lambda_{\min} = 0$, for which the minimum allowable distortion is achieved. Conversely, a safe maximum value is $\lambda = +\infty$ (to within the numerical limits of the hardware). This upper bound is however much too high in most cases and a better starting value can usually be inferred from the properties of the data; some examples will be illustrated in the next chapters.

2.2.4 Joint segmentation and allocation: dependent case

Let us now go back to the piecewise polynomial example: to model a polynomial, the family of data models will clearly comprise what we could call "polynomial prototypes", that is, set of polynomial templates indexed by their degree and whose parameters are estimated from the data samples they are applied to. Clearly, fitting such polynomial prototypes to different portions of the signal is no longer a pointwise process since the accuracy of the local polynomial parameters depends on the entire amount of data available to the underlying estimation process (think of Least Squares) while the rate stays constant; this, in turn, affects the cost functional in a nonseparable way. This is a common occurrence and, in fact, truly pointwise cost functionals are more the exception than the rule. Remember that the goal of optimal segmentation is to remove the fixed-window constraint, and windows are introduced in the first place to somehow discretize on a regular grid a given distortion function.

In general terms, nonseparable data models push back the optimal allocation problem to its original exponential complexity. Fortunately, however, *for models whose distortion is still additive over disjoint segments*, the framework of the previous section can be easily adapted using dynamic programming. This includes all data models for which

the classic fixed-window paradigm is applicable, leaving out only the recursive backward adaptive coding schemes.

Dependent allocation

Again, the sole hypothesis now is that the distortion function is additive over disjoint segments. We have chosen to tag the side information as a preamble to each segment using run-length coding, therefore the Lagrangian cost functional $J = D + \lambda R$ is itself additive over disjoint segments. If the segmentation was given a priori (a fixed-window splitting, for instance), then the modeling problem would become the simpler allocation problem illustrated above, where the data "points" to which the resources are assigned are just the segments themselves. For a N-point data vector there is only a finite number of possible segmentations, 2^{N-1} to be precise; if we let the segmentation be a free variable in the minimization process, we simply have to deal with a larger population of operational R/D points each of which is now indexed by segmentation-allocation pairs as in Figure 2.5. Again, if we choose to restrict the minimization to the convex hull of the composite set of points, we can solve the associated Lagrangian problem as a double minimization:

$$J^*(\lambda) = \min_{\boldsymbol{t}\in T} \min_{\boldsymbol{w}\in W(\boldsymbol{t})} \{J(\lambda)\}; \tag{2.17}$$

where T is the set of all possible segmentations. More in detail, a *segmentation* $\boldsymbol{t}$ is defined as a collection of $j+1$ time indices: $\boldsymbol{t} = \{t_0 = 1 < t_1 < t_2 < \ldots < t_{j-1} < t_j = N+1\}$, with j between 1 and N; the number of segments defined by any one $\boldsymbol{t}$ will be denoted by $\sigma(\boldsymbol{t})$, $1 \leq \sigma(\boldsymbol{t}) \leq N$, with the i-th segment being $x_{t_i}^{t_{i+1}-1}$. The notation for the allocation $\boldsymbol{w}$ must now reflects its dependence on the number of segments; in particular, $\boldsymbol{w}(\boldsymbol{t})$ is now a collection of $\sigma(\boldsymbol{t})$ model indices w_i, $1 \leq w_i \leq K$ and $W(\boldsymbol{t})$ is the set of all possible allocations for $\boldsymbol{t}$, with $|W(\boldsymbol{t})| = Q^{\sigma(\boldsymbol{t})}$.

The cost functional in (2.17) now involves two kinds of dependence, both in the rate and in the distortion; as we said, however, it is still additive over disjoint segments and therefore we can write:

$$J^*(\lambda) = \min_{\boldsymbol{t}\in T} \min_{\boldsymbol{w}\in W(\boldsymbol{t})} \{\sum_{i=1}^{\sigma(\boldsymbol{t})} d(t_i, t_{i+1}; w_i) + \lambda r(t_i, t_{i+1}; w_i))\} \tag{2.18}$$

where now the notation for the distortion and the rate highlights the beginning and the end of the data segment $x_{t_i}^{t_{i+1}-1}$ to which the model indexed by w_i is applied. Since data segments are now the basic allocation recipients in the optimization process, we don't need to separate the contribution of side information, which is absorbed into the rate requirements for a given segment; the notation $r(t_i, t_{i+1}; w_i)$ therefore includes here model index, model parameters (if applicable) and segment length.

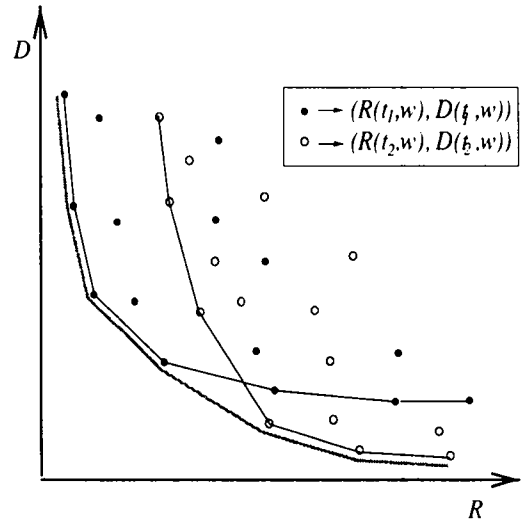


Figure 2.5: Composite convex hull.

Since all quantities are nonnegative, the inner minimization over $W(\boldsymbol{t})$ can be carried out independently term by term, reducing the number of comparisons to $K\sigma(\boldsymbol{t})$ per segmentation. Now the key observation is that, whatever the segmentation, all segments are coded at the same rate/distortion trade-off as determined by λ; therefore, for a given λ, we can determine the optimal $\boldsymbol{t}$ (in the sense of (2.17)) using once again a dynamic programming approach, although different in flavor. Just as in the previous case, the development of the dynamic algorithm can be illustrated formulating an optimality hypothesis. Suppose we know a breakpoint t^* belongs to the optimal segmentation $\boldsymbol{t}^*$; then it is easy to see that

$$J^*_{[1,N]}(\lambda) = J^*_{[1,t^*-1]}(\lambda) + \min_{\boldsymbol{t}\in T_{[t^*,N]}} \min_{\boldsymbol{w}\in W(\boldsymbol{t})} \{J(\lambda)\} \tag{2.19}$$

where subscripts indicate the signal range for the quantities involved (that is, $T_{[t^*,N]}$ is for instance the set of all possible segmentations for $x_{t^*}^N$). In other words, (2.19) states that if t^* is an optimal breakpoint, then the optimal cost functional for $x_{t^*}^N$ is independent of subsequent data. Again, this defines an incremental way to jointly determine the optimal segmentation and allocation as a recursive optimality hypothesis for all data points as segmentation breakpoints: for $1 \le t \le N$,

$$J^*_{[1,t]}(\lambda) = \min_{1\le\tau\le t} \{J^*_{[1,\tau-1]}(\lambda) + \min_{1\le k\le K} \{d(\tau,t;k) + \lambda r(k)\}\} \tag{2.20}$$

with $J^*_{[1,0]}(\lambda) = 0$ by definition. At each step t, we need to keep track of the new $J^*_{[1,t]}(\lambda)$ and of the minimizing τ only. The incremental process is illustrated graphically in Figure 2.6, where the first three steps of the algorithm are displayed; black lines represent

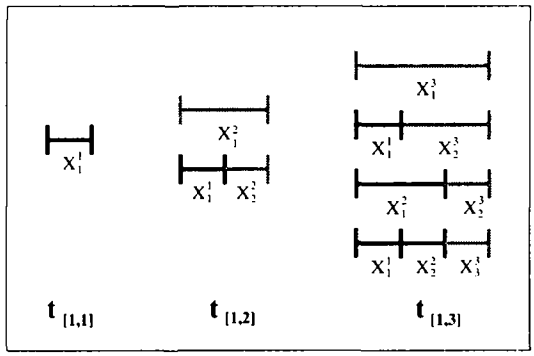


Figure 2.6: Incremental segmentation/allocation.

the sub-segmentations for which the minimum cost functional has already been computed while gray lines represent their extension to the current data length. It is clear how at step three, for instance, once we assume a breakpoint at $t = 2$, we don't need to explicitly compute the optimal segmentation for points x_1 and x_2, since it has already been determined at step two.

In terms of computational requirements, by looking at Equation (2.20) we can see that the minimization process at any step n entails the evaluation of n distinct cost functionals over segments from length 1 to n; as a general approximation, assuming that the distortion function for a n-point vector requires $O(n^\delta)$ operations, the total computational requirements for the algorithm are then on the order of $O(N^{\delta+2})$, while the storage requirements are $O(N)$.

Trellis-structured implementation

Computation of the cost functionals aside, the minimization in (2.20) requires exactly $N(N-1)/2$ comparisons for a given value of λ. If an iteration over the Lagrange parameter is required, as is often the case, it is much more convenient to split the algorithm in two parts: on the first pass all the rate and distortion values needed by equation (2.20) are computed and stored; subsequently, the minimum Lagrangian cost for a given λ can be determined at a lower computational cost, allowing for a faster iteration. This approach, which in computer science parlance would be called memoization, can be organized on a trellis; in contrast to the previous case, however, here the trellis functions more as a data structure than as an algorithmic device, since most states, whose number grows linearly with the time step, act only as place markers rather than as indicators of the encoder's state. Yet, the trellis structure offers an intuitive graphical representation of the minimization process and allows for all intermediate quantities to be conveniently organized;

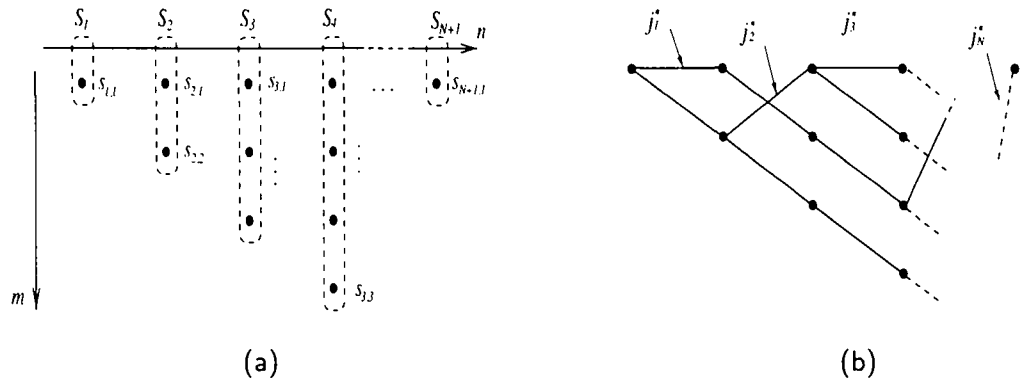


Figure 2.7: Trellis for the rate-dependent case:(a) trellis structure; (b) path population.

in most cases this leads to an improved overall performance, as the examples in the following chapters will show. The algorithm can be formalized as follows:

Algorithm 2.2: Trellis algorithm for the dependent case

Step 1: Initialization

Organize the data on a trellis (see Figure 2.7-(a)):

1 For $1 \leq n \leq N$
create a state set S_n containing n new states $s_{n,m}$ each, $1 \leq m \leq n$.

2 Then, for each $s_{n,m} \in \bigcup_n S_n$ (in some suitable order):
If applicable, associate to the state all local intermediate quantities for an incremental computation of the distortion.

Compute the distortion and rate values $d(n-m+1, n; k)$, $r(k)$ for $1 \leq k \leq K$ and associate these values to $s_{n,m}$.

3 Finally, create an extra state set $S_{N+1} = \{s_{N+1,0}\}$.

Step 2: Trellis path population

4 Select a value for λ; let $j_0^* = 0$.

5 For $1 \leq n \leq N$

Determine the minimum cumulative Lagrangian cost:

$$j_n^* = \min_{1 \leq m \leq n} \min_{1 \leq k \leq K} \{j_{n-m}^* + d(n-m+1, n; k) + \lambda(c + r(k))\}$$

using the values stored in the trellis.

Assume the minimum is for the pair (m^*, k^*): connect s_{n,m^*} to $s_{n+1,1}$ and associate (j_n^*, m^*, k^*) to the path. Also, connect $s_{n,m}$ to $s_{n-1,m-1}$ for $m > 1$ (see Figure 2.7-(b)).

Step 3: Backtracking

6 Obtain the optimal rate and distortion for the given λ from $j_{N+1}^* = d^*(\lambda) + \lambda r^*(\lambda)$.

7 If the rate constraint is met

Then follow the path backward in the trellis from $s_{N+1,1}$ to $s_{1,1}$ collecting, where applicable, the values for m^* and k^*. Assume j such values are collected, numbered from j to 1 as we proceed backward: the optimal segmentation is then $\boldsymbol{t}^* = \{t_0, \ldots, t_j, t_{j+1} = L\}$ with $t_i = t_{i+1} - m_i^*$ recursively, and the optimal allocation is $\boldsymbol{w}^* = \{k_1^*, \ldots, k_j^*\}$ (Figure 2.7-(b)).

Else proceed to the iteration over λ.

Step 4: Iteration over λ

See Step 4 in Algorithm 2.1.

The previous algorithm is very general and it applies to all cases of backward dependent segmentation for which rate and distortion are additive over disjoint segments. The details of the implementation obviously depend on the particular application and on the family of data models and in the examples of the following chapters this template will be suitably customized.

2.3 About side information

In Section 2.2.3 we have shown that information about the allocation and the segmentation is an integral part of the minimization process, with the major consequence of creat-

ing dependence between data points with respect to the chosen cost functional. However, an issue has been so far swept under the rug, so to speak, and that is the actual size of this additional information; it is clear that, just as with all other quantities, the final result of the allocation process depends on the price (in bits) paid to describe the allocation itself.

We find ourselves in a self-referential situation: the allocation depends on the amount of side information, and the amount of side information depends on the allocation. The approach we will follow in the practical coding schemes of the next chapters is to cut open this circularity by fixing the price of side information a priori. Yet, a more exhaustive approach would allow for different possible ways to encode the allocation structure, much in the same fashion as we allow for different data models, and would perform a double optimization over both sets. We will now try to explore what possibilities lie behind the side information issue, both from a theoretical and a practical point of view.

2.3.1 Upper and lower bounds

The total amount of side information for each segment can often be split in two components: a first portion is devoted to encoding the model index and possibly its related parameters, if these are adaptive; the second portion encodes the start- and end-point of the segment (or simply its length). The first portion of the side information content is basically fixed, depending as it does on the family of models, and can be thought of as an integral part of the rate requirements of a given model; we will not consider it further. The second portion, however, has a subtler nature. Denote by $\hat{x}_1^{R_0}$ the binary stream after the data have been encoded; this stream comprises the encoded data and the side information related to the segmentation. We can always choose to lump the segmentation description in a prefixed data block so that the output stream is partitioned as $\hat{x}_1^{R_0} = [s_1^S; y_1^R]$; the fisrt part is the side information (S bits) and the second part, with $R = R_0 - S$, is the actual data.

In order to run the allocation process, we must fix in advance the amount of side information associated to a segment. We can identify two extreme cases. If we have no idea at all on the final number of segments, we must assume that any of the segments can be as long as the output stream itself; therefore we must set the cost of side information to $\log_2 R_0$. At the other extreme, we could consider the completely unrealistic situation in which we know exactly how many segments we will have in the end; say this number is M. In this case, we can derive a lower bound on the cost of side information as follows: associate to y_1^R a length-R binary vector z_1^R; a value of one for z_n means that y_n is the startpoint of a data segment, and there will be M ones in the vector. We can therefore encode z_1^R with no less than $RH(M/R)$ bits, where $H(\cdot)$ is the binary entropy function:

$$H(p) = -p \log_2 p - (1-p) \log_2 (1-p). \tag{2.21}$$

From the relation

$$R_0 = R(1 + H(M/R)) \tag{2.22}$$

we can determine the value for R and set the side information cost per segment to $(R/M)H(M/R)$, which is the minimum possible.

2.3.2 Coloring bits[3]

So far it seems that, for lack of better knowledge, all we can do is use the $\log_2 R_0$ estimate since any value smaller than this does not guarantee we can encode *any* possible segmentation. In fact it is not so; we can actually set the cost of side information to any value less than $\log_2 R_0$ without making any assumption on the number of segments and still be able to encode all segmentation structures. In so doing, however, we will have to pay a fixed extra price for the added service.

The idea is to blend the two sequences y_1^R and z_1^R into a single one; we could look at this blending as (somewhat picturesquely) at a process where the bits in $\boldsymbol{y}$ which correspond to the beginning of a segment are "colored" (red, for instance...); information theorists would call this, more seriously, a stream punctuating scheme. Colored bits are equivalent to a third symbol, besides zero and one; since the original stream is binary, allowing for an extra symbol corresponds to a data expansion which depends on its implicit probability, and this is the fixed price mentioned above. Similarly, a colored bit must be binary encoded at a cost of, say, c bits; this will be the cost we select for the side information of a segment. If we denote by ρ the data expansion factor caused by the presence of the third symbol, we can write the following lower bound for the final output size:

$$R(1 + \rho) + Mc \geq R(1 + H(M/R)) \tag{2.23}$$

where the expression on the left represents the final length of the binary stream (R_0) after using an arbitrary number M of colored bits, and the expression on the right is the lower bound for M segments which we derived previously. We have therefore a relation between the cost of a colored bit and the fixed expansion factor we have to allow for; from

$$c \geq (R/M)(H(M/R) - \rho) \tag{2.24}$$

we can minimize the right hand side with respect to all meaningful ratios M/R (that is, $0 \leq M/R \leq 1/2$ [4]) and obtain either a lower bound for the cost of a colored bit:

$$c \geq \log_2 \left(\frac{1}{2^\rho - 1} \right) \tag{2.25}$$

[3] For the discussion in this section I am greatly indebted to Claudio Weidmann; in particular, at my insistent request, he single-handedly derived the bound in (2.25) over a pizza and a beer.

[4] For $M/R > 1/2$ we simply switch assumptions, and assume each bit is an independent segment *unless* a colored bit is used.

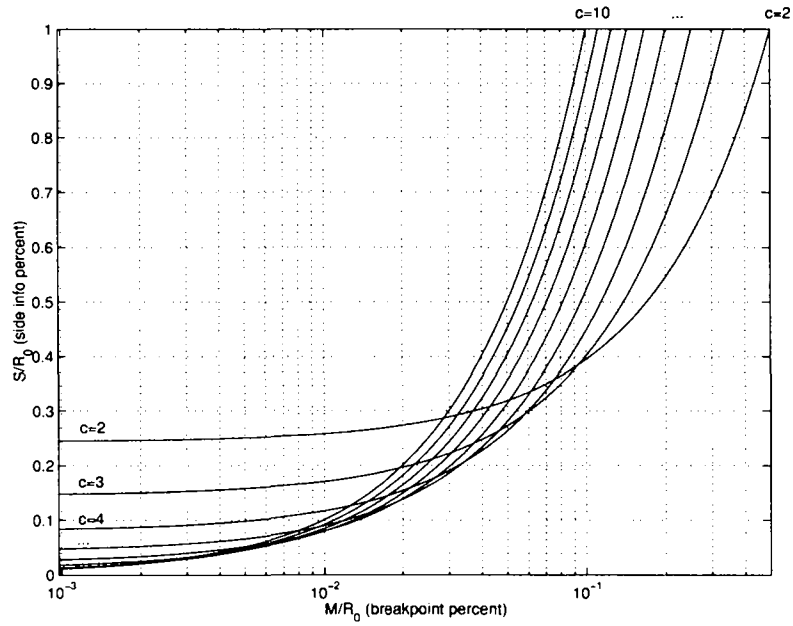


Figure 2.8: Percentage of side information as a function of breakpoint density.

or for the expansion factor given the size of side information:

$$\rho \geq \log_2(2^{-c} + 1). \tag{2.26}$$

With these bounds we can derive an expression for $p(M/R_0) = S/R_0$, the percentage of output bits spent on side information as a function of the density of breakpoints. We have

$$p_e(M/R_0) = \frac{M}{R_0} \log_2 R_0 \tag{2.27}$$

for the "easy" signaling scheme and

$$p_c(M/R_0) = \frac{M}{R_0} \frac{c}{1+\rho} + \frac{\rho}{1+\rho} \tag{2.28}$$

for the colored bits scheme. The curves corresponding to $p_c(\cdot)$ are displayed in Figure 2.8, parametrized in c, for $R_0 = 2^{10}$; since ρ approaches zero as c approaches $\log_2 R_0$, each of the curves also approximately represents the function $p_e(\cdot)$ when $R_0 = 2^c$. The value of c can therefore be used as a parameter in the optimization: the cost of side information is set to c and, correspondingly, the rate requirements for the models are augmented by a

factor of ρ; an iteration over c yields the optimal solution. The gain for this extra adaptivity is given by the offset between p_c and p_e; for a breakpoint density lower than 1% of the total number of points, for instance, the maximum achievable gain is around 20%.

Implementation

We now know the bounds on the performance of colored bits, but how do we actually color them? Building a universal punctuation scheme which achieves the bound in (2.23) is, to the best of our knowledge, an open question. We can however implement the coloring mechanism quite simply using an arithmetic coder, with a performance remarkably close to the bound.

Arithmetic coding will be discussed in detail in Chapter 4; for the time being, all we need to know is that, given a k-ary input alphabet and an associated set of probabilities p_k, an arithmetic coder compresses an N-element string of input symbols to a binary word of length $\log_2 1/A$, with

$$A = \prod_{k=1}^{N} p_k^{(\#\text{ of } k\text{'s in the input})} \tag{2.29}$$

We will assume that the data vector y_1^R is a Bernoulli 1/2 sequence; this is reasonable since we are in a compression context and the data models should output maximum entropy sequences. If we process the data vector with a binary arithmetic coder, we obtain

$$A = p_0^{h(\boldsymbol{y})}\, p_1^{(R-h(\boldsymbol{y}))} \tag{2.30}$$

where $h(\boldsymbol{y})$ is the Hamming weight of the sequence. By setting $p_0 = p_1 = 1/2$, as by hypothesis, we obviously have $\log_2 1/A = R$, which is consistent with our assumption of a maximum entropy sequence. Now we introduce colored bits, which are represented as a third symbol of arbitrary probability ϵ. We still want the "uncolored" data bits to be equiprobable; the admissibility condition then imposes $p_0 = p_1 = (1-\epsilon)/2$ and with this new probability assignment we encode the data to obtain

$$A_\epsilon = \left(\frac{1-\epsilon}{2}\right)^R; \tag{2.31}$$

from this the expansion factor is derived as:

$$1 + \rho = \log_2 \frac{2}{1-\epsilon}. \tag{2.32}$$

A colored bit will cost us $\log_2 1/\epsilon$ bits, as we can see from (2.29) so that in the end we have:

$$\rho = \log_2 \frac{1}{1-2^{-c}}. \tag{2.33}$$

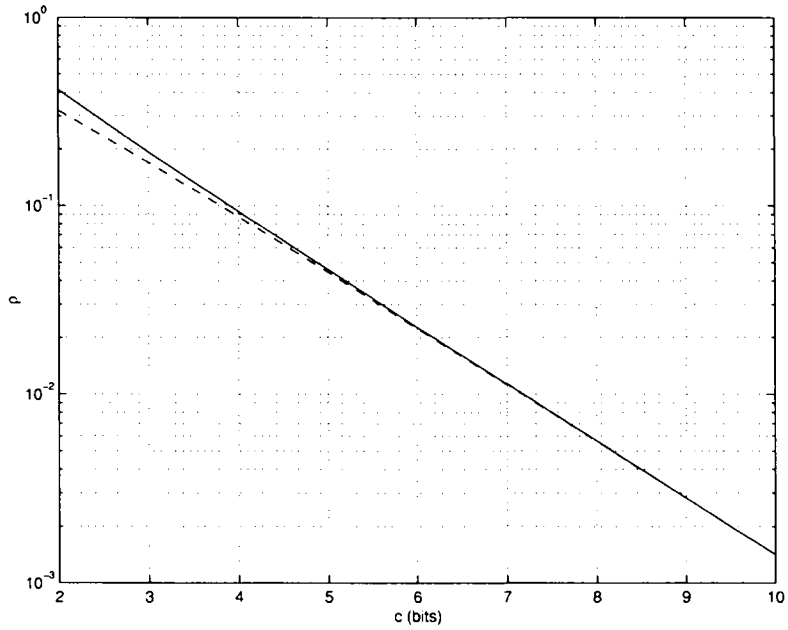


Figure 2.9: Expansion factor ρ vs. cost of a colored bit: theoretical upper bound (solid) and arithmetic coding performance (dashed).

We can compare this relation to the theoretical bound in (2.26); the curves are displayed in Figure 2.9, and they are almost indistinguishable for even moderate values of c.

Is it worth it?

Optimizing the segmentation with respect to the cost of side information requires an iteration of the segmentation/allocation process over at most $\log_2 R_0$ values for c and can be seen as an additional refinement step. In most cases either the "easy" choice of $c = \log_2 R_0$ or a guess on the potential number of segments (and therefore on the corresponding c) proves adequate; in applications where the premium is entirely on a low bitrate, the iteration might be a need. One final remark is however necessary: in many compression applications (amongst which the polynomial modeling to be discussed in the next chapter) it is difficult to exactly split the side information between a segmentation structure component and a model parameters component. It is easy to see that in the dependent allocation case the range of application of a given model (i.e. the segment's length) is hardly separable from the other segmental parameters. In this case the optimization of the seg-

mentation with respect to the side information rests entirely with a careful design of the family of data models.

2.4 Summary

Piecewise stationary signals can be effectively analyzed by a sequence of local models fitted to the different portions of the signal itself; the implicit time segmentation, together with the sequence of coding models, can be determined by formulating the fitting process as a generalized rate-distortion problem and the result is optimal with respect to the chosen family of models and the chosen cost function. Section 1 reviewed the connection between rate-distortion theory and the problem at hand. Section 2 introduced the fundamental building blocks of the optimal allocation problem and discussed in detail two fundamental cases of segmentation, for independent and dependent cost functions. An efficient implementation can be arrived at using Lagrange multipliers, which conveniently bridge the two end cases of signal modeling and signal compression, and general trellis-based algorithmic procedures have been described; these procedures will be used as templates in the following chapters, where specific coding problems are taken into account. The issue of side information is also very important, since the optimal segmentation and allocation are strictly data-dependent, and their description is an integral part of the coded data. This is tackled in Section 3 and the idea of "colored bits", used to encode a segmentation structure at an arbitrary cost, is also discussed. Finally, Appendix A looks at the allocation problem from a different perspective: the trellis algorithm is used to explore all possible rates at once. This approach is usually too costly to be truly practical, but is nonetheless rather interesting in some of its slightly suboptimal heuristic variations.

Appendix 2.A Alternative to the iteration

As we have seen, the use of Lagrange multipliers casts the search for the optimal allocation into an efficient algorithmic framework. Yet, one could argue, this approach is not without drawbacks. First and foremost, some R/D points, although optimal, are not reachable since they are off the convex hull, as we already pointed out in section 2.2.2. Then, as with all iterative algorithms, the execution time is variable, depending as it does on the initial guess for λ. Furthermore, each minimization provides the optimal allocation only for a single total rate and the minimization plus iteration process must be repeated if a different rate is desired. Finally, a new global iteration is needed if the data vector is extended with new points.

Part of these problems were addressed in [47], where the GBFOS algorithm is used to obtain the optimal allocation for all possible rates simultaneously in the case of independent quantization of several sources. The scheme is however designed only for a set of quantizers with contiguous rates and no provisions are made for side information or data vector extensions. A more general approach which can account for both these issues makes use (once again) of dynamic programming; while this is no new tool in this context (dating back at least to 1980 [59]), it has not been widely used because of its purported computational complexity. Indeed, it is a costly approach, but it solves the aforementioned problems and, as we will show, is amenable to practical algorithmic implementations which trade computational complexity for global optimality at a linear cost in the number of data points.

2.A.1 Independent case

Let us initially consider the dynamic programming solution for the case of independent allocation, no side information (see Section 2.2.2). For any given bit budget R_0 recall that the optimal allocation $\boldsymbol{w}^*$ is defined as:

$$\boldsymbol{w}^* = \arg \min_{\boldsymbol{w}|R(\boldsymbol{w})=R_0} \{D(\boldsymbol{w})\}. \tag{2.34}$$

Due to the additivity and non-negativity of rate and distortion values we can write for any data subset x_1^n:

$$\min_{w_1^n|R(w_1^n)=R} \{D(w_1^N)\} = \tag{2.35}$$
$$\min_{1\le k\le K} \{ \min_{w_1^{n-1}|R(w_1^{n-1})=R-r(n;k)} \{D(w_1^{n-1})\} + d(n;k)\}$$

where rate and distortion values are computed over the implicit data subset. In other words, the optimal solution at step n with rate R can be obtained from the minimum partial solution at step $n-1$ with rate "one model away" from R. The relation is clearly

recursive in n and valid for all values of R; this leads to the following trellis-based algorithm[3]: let S_n be the set of meaningful states in the trellis at step n; each element of S_n is a (r, d, k) triple where r is the cumulative rate, d is the cumulative distortion and k is the last model index; each (r, d, k) represents the value of the inner minimization in (2.35) for rate r. Then the algorithm is the following:

Algorithm 2.3: Dynamic programming: independent case

Step 1: Initialization

1 Let $S_0 = \{(0, 0, 0)\}$.

Step 2: Allocation

2 At each step n, $1 \leq n \leq N$

let $S_n = \emptyset$;

precompute $d(n; k)$ for $k = 1, \ldots, K$;

for all previous states $(r, d, k) \in S_{n-1}$

for $k = 1$ to K

compute the new cumulative rate $r' = r + r(n; k)$

compute new cumulative distortion: $d' = d + d(n; k)$

(*new state:*)

look for a state (r', δ, κ) in S_n (for any δ and κ)

if there's no such state

then $S_n = S_n(r', d', k)\}$

else if $d' < \delta$ (*pruning*)

then $S_n = (S_n \setminus \{(r', \delta, \kappa)\})(r', d', k)\}$

connect old state to new state.

[3] Please note that, although the trellis notation which follows resembles that of the previous sections, the algorithms are unrelated; here the states represent the cumulative rate, while in the previous cases the states were associated either to the model index or to the length of the data segments.

Step 3: Backtracking:

3 Select $(r, d, k) \in S_N$ with $r = R_0$ (or r closest to R_0).

4 Collect the model indices following back along the branches in the trellis;

Please note that in the end not only do we have the optimal allocation for all rates, but also the optimal allocation for all data subsets $x_1^n, 1 \leq n \leq N$; extending the optimal allocation to a longer data vector is therefore straightforward.

The computational requirements of the algorithm can be estimated as follows. Besides the computation of the R/D pairs, at each step the number of trellis operations (sums, comparisons) is proportional to the number of states $|S_n|$. A simple estimate for the latter is obtained observing that, at each step, the number of different states is equal to the number of reachable rates. The smallest and largest such rates increase at each step by $r_{\min}$ and $r_{\max}$ respectively, where $r_{\min}$ and $r_{\max}$ are the minimum and maximum rates amongst all items in the family of data models. The number of reachable states at each step is smaller if all possible rates share a common factor. The number of states at step n is therefore upper bounded by

$$|S_n| \leq n \frac{r_{\max} - r_{\min}}{\mathrm{GCD}_k\{r(\cdot, k)\}} + 1 = n\Delta + 1. \tag{2.36}$$

Summing across all points for a length-N data vector, storage and computational requirements are therefore proportional to $(\Delta/2)(N^2 + N) + N$.

2.A.2 Partially independent case

As usual, we choose to encode the structure of the allocation by associating side information to model transitions. Suppose the cost of signaling a transition is c bits; now applying model k to the data point x_n leaves the distortion as before but changes the effective rate requirement to $r(n; k) + c$ if the model for the previous data point was not the k-th as well. Integrating this dependence in the previous dynamic programming framework determines only an increase of the state space. Line 2 of the previous allocation algorithm can be modified as follows:

Algorithm 2.4: Dynamic programming: partially independent case

...

2 At each step n, $1 \leq n \leq N$

let $S_n = \emptyset$;

precompute $d(n;k)$ for $k = 1, \ldots, K$;

for all previous states $(r, d, k) \in S_{n-1}$

for $k = 1$ to K

compute the new cumulative rate: if $h = k$

then $r' = r + r(n;k)$

else $r' = r + r(n;h) + c$

compute new cumulative distortion: $d' = d + d(n;k)$

(*new state:*)

look for a state (r', δ, κ) in S_n (for any δ *only*)

if there's no such state

then $S_n = S_n(r', d', h)\}$

else if $d' < \delta$ (*pruning*)

then $S_n = (S_n \setminus \{(r', \delta, h)\})(r', d', h)\}$

connect old state to new state.

Note that now we are allowed to prune only between states with the same rate *and* the same model index. The increase in state space size is due to the fact that the rate range at each step is augmented by c bits and to the fact that states with the same rate but different quantizer indices are now distinct, for an overall K-fold increase. The upper bound in (2.36) remains formally valid provided we use:

$$\Delta = \frac{K(r_{\max} - r_{\min} + c)}{\mathrm{GCD}_k\{r(\cdot;k), r(\cdot;k) + c\}} \tag{2.37}$$

Although in this description we have represented the cost of side information as a constant for simplicity, the very same algorithmic procedure can be used if the cost of a transition is $c(i,j)$, where i and j are the indices of contiguous models. If some prior infor-

mation is available about the data which makes the use of a quantizer more likely than others, this information can be usefully incorporated in $c(i,j)$ to minimize side information.

2.A.3 Dependent case

When the cost function is non-additive the situation becomes more complex in that, at each step n we have to keep track of data segments that are still "open". Indeed, consider this simple example: assume we have just two models, A and B; at $n=1$ we have only two well-defined states, corresponding to coding the first data point as a complete segment of length one with either model A or model B. At the next step, however, there are six states amongst which we can prune: the first four correspond to splitting x_1^2 into two length-one segments (AA, AB, BA, BB), while the other two correspond to coding x_1^2 with a single model, either A or B. We can tackle the situation by introducing a fictitious model index $k=0$ with no associate distortion and zero rate, which acts as a marker for open segments. Unfortunately, it turns out that in this case the number of states grows quadratically with n, for a total number of states in the trellis proportional to N^3. This is usually too memory intensive for the algorithm to be of practical value. For completeness, here is however the modified algorithm

The aggregate set of segmentations and relative allocations has cardinality $(K+1)^N$; indeed, for $1 \le i \le N$, there are $\binom{N}{i}$ different segmentations with exactly i segments and each of these admits K^i different allocations. We can see this fact from another angle by introducing an extra model index $k=0$ and representing a segmentation/allocation pair in a joint way as a vector $\boldsymbol{z}$, with the convention that leading zeros to a model index represent the number of consecutive points the following model is applied to. For instance, $\boldsymbol{z} = \{0,0,A,0,B,B,0,A\}$ corresponds to $\boldsymbol{t} = \{1,4,6,7,9\}$, $\boldsymbol{w} = \{A,B,B,A\}$. Clearly, $|Z| = (K+1)^N$. We call an allocation "open" if at least its last element is zero. With this notation, equation (2.35) becomes

$$\min_{z_1^n | R(z_1^n)=R} \{D(z_1^n)\} =$$

$$\min_{1\le k\le K} \{ \min_{1\le m\le n-1} \{ \min_{z_1^m | z_{m+1}^{n-1}=\mathbf{0}, R(z_1^m)=R-r(m,n;k)} \{D(z_1^{n-1})\} + d(m,n;k)\} \}. \qquad (2.38)$$

What this expression means is that, at step n, the partial allocations "one model away" include also all open allocations pointing back to an earlier step than $n-1$; these are the z_1^n for which the model index is zero from some step m to n for all $m < n$. We can still associate a state to each possible value of the inner minimization and modify the trellis algorithm as follows:

Algorithm 2.5: Dynamic programming: dependent case

Step 1: Initialization

1 Let $S_0 = \{(0,0,0,0)\}$.

Step 2: Allocation

2 At each step n, $1 \leq n \leq N$
let $S_n = \emptyset$;
for all previous states $(r,d,k,l) \in S_{n-1}$
$S_n = S_n(r,d,0,l)\}$, and connect old state to new state;
for $k = 1$ to K
compute new cumulative rate: $r' = r + r(l,n;k) + c$
compute new cumulative distortion: $d' = d + d(l,n;k)$

(*new state:*)
look for a state $(r',\delta,\kappa,\lambda)$ in S_n for $\kappa \neq 0$ and any δ and κ
if there's no such state
then $S_n = S_n(r',d',k,n)\}$
else if $d' < \delta$ (*pruning*)
then $S_n = (S_n \setminus \{(r',\delta,\kappa,\lambda)\})(r',d',k,n)\}$
connect old state to new state.

Step 3: Backtracking:

3 Select $(r,d,k,l) \in S_N$ with $r = R_0$ (or r closest to R_0) and collect the model indices (including the zeros) following back along the branches in the trellis.

Now we can prune between states with the same rate only if the corresponding allocations are not open, while we must extend all previous states to an equivalent number of open allocations. The number of states can be estimated as follows: at step n there are o_n open allocations and q_n closed ones; the total number of states is $S_n = o_n + q_n$.

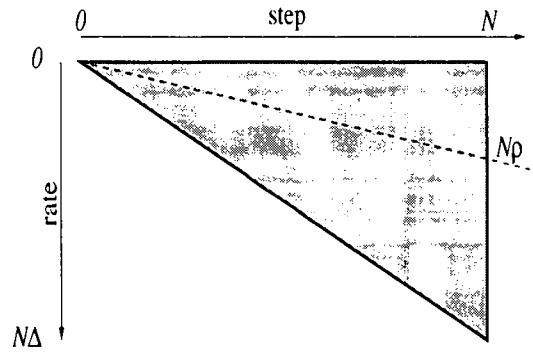


Figure 2.10: Continuous approximation for the trellis.

Closed states have a well-defined rate, and therefore the bound in (2.36) is still valid with $\Delta = r_{\max}/\mathrm{GCD}_k\{r(\cdot,\cdot;k)\}$ (remember that now the minimum rate is zero, because of open allocations). Open states are however as many as the states at the previous step: $o_{n+1} = S_n$; this leads to $S_n = S_{n1} + n\Delta + 1$, or $S_n = \Delta n(n+1)/2 + n$. Summing over n leads to the anticipated result of a cubic number of states. Because of this, we will not pursue this algorithmic line any further.

2.A.4 Implementation Issues

In the following section we will illustrate some implementation strategies which reduce the complexity of the algorithm in most practical cases. To facilitate the discussion, we will make use of a graphical representation of the trellis based on a *continuous approximation*. First of all observe that the model rates can be normalized so that they are coprime and that the smallest rate is zero. In this case at each step n the minimum rate is zero and the maximum rate is $n(r_{\max} - r_{\min}) = n\Delta$. With this normalization, the trellis can be represented graphically as a wedge (see Fig. 2.10): the step number n runs along the x-axis while the rate r is plotted along the y-axis (pointing downwards); the slope of the wedge is simply Δ. In the continuous approximation we consider n and r as real valued quantities; this allows us to infer estimates on the number of states as surface measures: the number of states in Fig. 2.10 is $N^2\Delta/2$, which, for a moderately dense set of admissible model rates, is very close to the bound in (2.36) summed over all steps.

Zooming in

In most compression applications, we are concerned with one particular bit rate or at most with a narrow range of rates around a given bit budget. The trellis structure allows us to

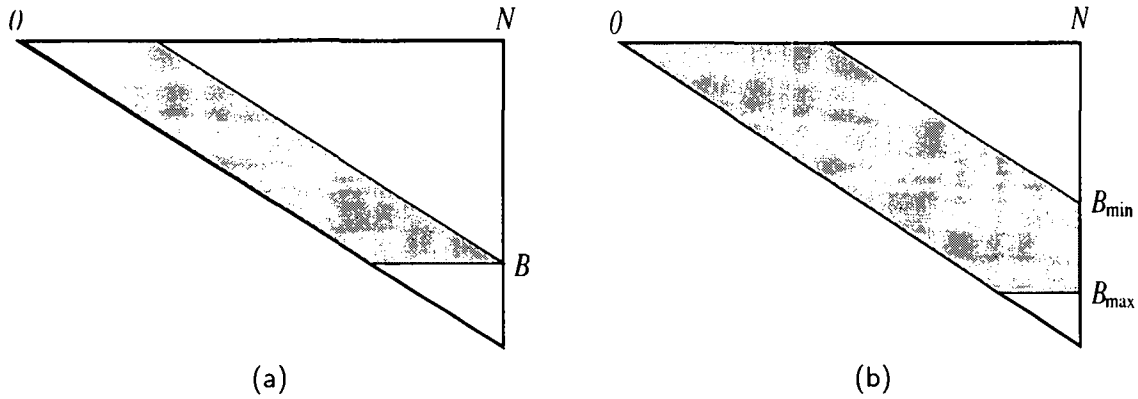


Figure 2.11: Zooming in: (a) single rate; (b) rate interval.

"zoom in" to the range of interest, reducing the total number of trellis states (and of operation) to $O(N)$. Suppose we are interested in bit rate of R_0 at the final step N. At each intermediate step n, it is enough to keep only those states whose rate r satisfies

$$R_0 - (N-n)\Delta \leq r \leq R_0; \tag{2.39}$$

all paths passing through any other state will either undershoot or overshoot the target rate. This is represented in Fig. 2.11-(a); the total number of meaningful states for the trellis is proportional to the shaded area A in the wedge, yielding

$$A = R_0(N - R_0/\Delta). \tag{2.40}$$

Similarly, if we zoom in on a range of rates between $R_{\min}$ and $R_{\max}$ (see Fig. 2.11-(b)), the number of states is proportional to

$$A = R_{\max}(N - R_{\min}/\Delta). \tag{2.41}$$

Block-by-block and continuous coding

Two classes of applications require us to terminate the trellis search before the global optimum is found; this happens when either the size of the data vector size is unknown a priori or it imposes storage requirements which are too large, or when there is a limit on the processing delay before quantization and encoding of data points. In both cases, the rate requirement can only be formulated in terms of an average bitrate of ρ bits/symbol, with $r_{\min} \leq \rho \leq r_{max}$; in the graphical representation the target bitrate becomes a line of slope ρ, as in Fig. 2.10.

A first approach is to partition the data into size-L contiguous blocks and run the "zoom in" algorithm separately for each block for a target rate $R = \rho L$. The resulting trellis configuration is displayed in Fig. 2.12-(a).

A second approach, more in line with standard trellis practice, is the following. Assume we start building the trellis up to $n = L + 1$, at which point we backtrack L steps and find the *locally optimal* path for rate $R = (L + 1)\rho$. For a sufficiently large L, we can assume that $s_1 = (r_1, d_1, k_1)$, the initial state of this path at $n = 1$, is actually the initial state of the *globally optimal path* and we can encode and send the associated first data point using model k_1. If s_1 is indeed optimal, the globally optimal path will unwind inside a wedge starting at s_1 even if the globally optimal and the locally optimal path afterwards differ in all states but the first one. This means that at step $L + 2$ we need only update the states in S_{L+1} whose rate r satisfies

$$Lr_{\min} \leq r - r_1 \leq Lr_{\max} \tag{2.42}$$

which requires on the order of L operations and generates $L\Delta + 1$ new states. At the next step the process is repeated, and in general, at step $n > L$ we backtrack L steps from rate $R_n = n\rho$, encode the point with model number k_{n-L}, associated to state s_{n-L}, and update the subset of states in S_n for which $Lr_{\min} \leq r_n - r_{n-L} \leq Lr_{\max}$. Finally, to ensure stability we only need to show that at each step the target rate R_n is within the new reduced set of states S_n. This can be shown by induction: assume the proposition holds at step n; we backtrack and find the locally optimal rate r_{n-L}. Now, it must be

$$r_{n-L} + Lr_{\max} \leq B_n \leq r_{n-L} + Lr_{\min}, \tag{2.43}$$

otherwise R_n would not be reachable in L steps from r_{n-L}. At step $n+1$ it is $R_{n+1} = R_n + \rho$ and since $r_{\min} \leq \rho \leq r_{\max}$, it is also

$$r_{n-L} + Lr_{\max} + r_{\max} \leq R_{n+1} \leq r_{n-L} + Lr_{\min} + r_{\min} \tag{2.44}$$

and it is immediate to recognize in the two outer terms of the inequality the maximum and minimum rates in S_{n+1}.

The storage and computational costs of the algorithm are clearly linear in n. A graphical representation of the resulting "tiled wedges" is displayed in Fig. 2.12-(b). For both strategies, the resulting suboptimality is heavily dependent on L. In the block by block approach, the bitrate requirement is followed exactly over length-L segments; the price we pay is a high distortion if the block size is small compared to the rate of change of the signal. The continuous coding approach follows the optimal path more closely but usually does not yield a constant bitrate over fixed blocks. Both methods also introduce "out of budget" side information; while in the block algorithm however this happens only at each block boundary, in the continuous algorithm continuity of states is not preserved

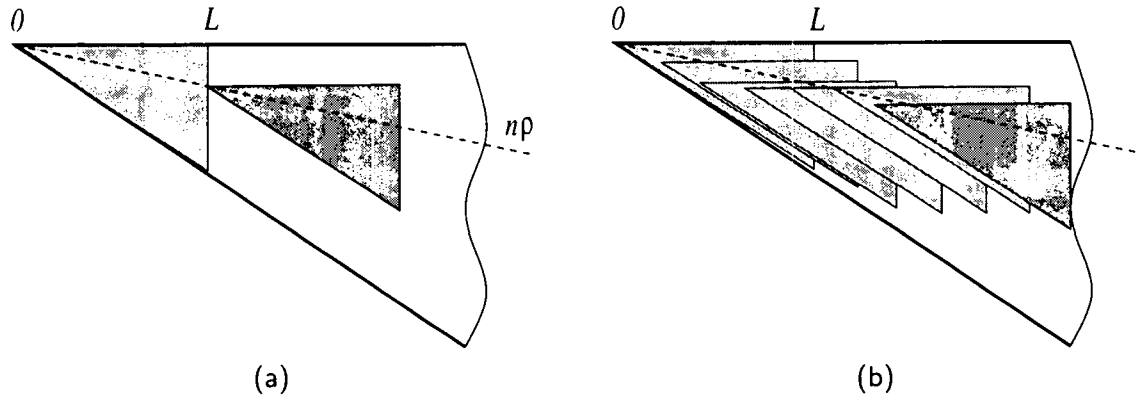


Figure 2.12: Continuous coding: (a) block by block; (b) backtracking.

and the price can be substantially higher. The best tradeoff is obviously dependent on the quantization problem and on the data, and it must be necessarily tested "on the field". A numerical example is displayed in Fig. 2.13: a synthetic 500-point data vector is generated by an iid random number generator whose variance switches between α and β according to a Poisson process with $\lambda = 0.01$. Two quantizers, 8 and 16 bits, are matched to α and β and the cost of side information is 8 bits. The solid line represents the globally optimal path for a target bitrate of 4 bits/point, while the circles and the diamonds represent the block and continuous coding solutions for $L = 150$. Marks on the upper line indicate the switch points in the data source.

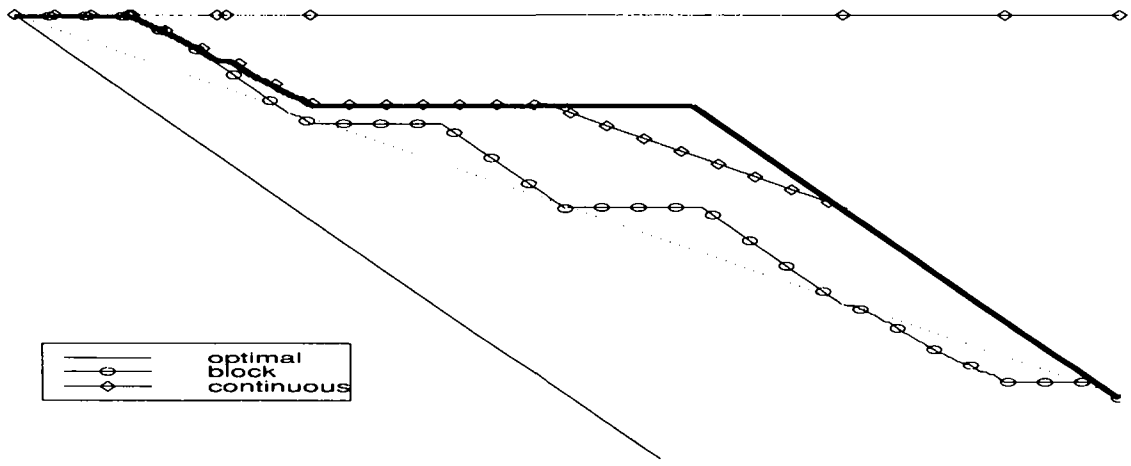


Figure 2.13: Experimental example: block by block (circles) and continuous (diamonds) coding solutions.

Chapter 3

Local Polynomial Modeling

Sicelides Musae, paulo maiora canamus!
– P. VERGILIUS MARO, *Eclogae, IV*

Ne sutor supra crepidam.
– Latin proverb

In the previous chapter, we have introduced the example of piecewise polynomial functions as a convenient example of a piecewise stationary signal. Indeed, the local polynomial approximation of real functions is a very well studied topic, especially in the form of spline approximation [12]. Splines are parametrized polynomial functions which are used to interpolate a given function over a set of selected points (called knots); higher-degree splines allow for an increasing order of continuity between juxtaposed pieces, with correspondingly increasing approximation smoothness, and several techniques exist to determine a good set of interpolation knots. These approximation techniques, however, are usually developed in what we could call a "rate-indifferent" framework, in the sense that the goal is the minimization of the distortion only (for an exception, see [33]). In the case of spline approximation, for instance, the number of knots (which corresponds to our segmentation breakpoints) is a pre-determined parameter whose direct relation to the overall accuracy does not find a parallel expression in terms of overall rate. Similarly, the inevitable quantization of the polynomial coefficients themselves is usually viewed as an additional, separate step; this may be acceptable when the ultimate goal is data modeling and

the (implicit) rate is high; but it leads to appreciably suboptimal results if the purpose of the modelization is data compression and the rate is severely constrained. In our dynamic allocation scheme, however, both these goals are seamlessly connected by the Lagrange parameter, which acts as a knob between modelization (infinite rate) and compression (minimal rate), and in all cases the solution is optimal with respect to the distortion measure.

In this chapter we will study the piecewise polynomial case in greater detail, both as a convenient testbed for the algorithmic techniques introduced so far and as an important approximation problem in its own right. Indeed, piecewise polynomial functions are a special case of piecewise smooth functions, which are interesting mathematical objects used to model a great variety of natural phenomena. In particular, we would like to compare the rate-distortion behavior of local polynomial modeling based on dynamic segmentation to nonlinear wavelet approximation. Wavelets have long been considered ideal candidates for piecewise smooth function due to their vanishing moments properties [27], and the question now is to see how these properties carry through in an operational rate-distortion scenario. In order to perform this comparison, we will first take a brief detour into the realm of continuous-time functions and derive two R/D upper bounds for the approximation methods we intend to study; we will then discretize the problem in order to obtain experimental results which we will compare to the theoretical bounds.

3.1 Wavelets and compression: a brief overview

Wavelets [61, 27, 57] have had an important impact on signal processing theory and practice. In particular, wavelets play a key role in compression, image compression being a prime example. This success is linked to the ability of wavelets to capture efficiently both stationary and transient behaviors. In signal processing parlance, wavelets somehow avoid the fixed-window problem (as in the short-time Fourier transform, for example), since they work with many windows via their scaling property. In the case of piecewise stationary processes wavelet methods are possible models as well, since they are able to both fit the stationary part and capture the breakpoints. In particular, in the context of smooth functions, the *vanishing moments* property is of particular interest; a wavelet $\varphi(t)$ is said to have $G+1$ vanishing moments if

$$\langle \varphi_n(t), t^m \rangle = 0 \quad \text{for } 0 \leq m \leq G, \tag{3.1}$$

where $\langle \cdot, \cdot \rangle$ is the inner product defined on L_2. This means that, for a piecewise polynomial function, the only nonzero wavelet coefficients are those associated to the breakpoints of the function.

When one is interested in data compression, a key question is not just the approximation behavior, but the effective rate-distortion characteristic of schemes where wavelets

and scaling functions are used as elementary approximation atoms. Indeed, compression is the trade-off between description complexity and approximation quality; given an object of interest, or a class of objects, one studies this trade-off by choosing a representation (e.g. an orthonormal basis) and then deciding how to describe the object parsimoniously in the representation: such a parsimonious representation typically involves an approximation. For example, for a function described with respect to an orthonormal basis, only a subset of basis vectors might be used (subspace approximation) and the coefficients used in the expansion are always approximated via quantization. Thus, both the subspace approximation and the coefficient quantization contribute to the approximation error. More formally, for a function f in $L_2(\mathbb{R})$ for which $\{\varphi_n\}$ is an orthonormal basis, we have the approximate representation.

$$\hat{f} = \sum_{n \in I} \hat{\alpha} \varphi_n \tag{3.2}$$

$$\hat{\alpha}_n = Q[\langle \varphi_n, f \rangle] \tag{3.3}$$

where I is an index subset and $Q[\cdot]$ is a quantization function, such as for example the rounding to the nearest multiple of a quantization step $\triangle$:

$$\hat{\alpha} = Q[\alpha] = \triangle \cdot \left(\left\lfloor \frac{\alpha}{\triangle} \right\rfloor + \frac{1}{2} \right). \tag{3.4}$$

Typically, the approximation error is is a squared error measured by the L_2 norm:

$$\epsilon = \|f - \hat{f}\|_2^2. \tag{3.5}$$

The description complexity corresponds to describing the index set I, as well as describing the quantized coefficients $\hat{\alpha}_n$. The description complexity is usually called the rate R, corresponding to the number of necessary binary digits. Therefore the approximation $\hat{f}$ of f leads to a rate-distortion pair (R, ϵ), indicating one possible trade-off between description complexity and approximation error. This example, despite its simplicity, is quite powerful and actually used in practical compression standards; it however raises the following questions naturally:

A: What are the classes of objects which are of interest and for which the rate-distortion trade-off can be well understood ?

B: If approximations are carried out in bases, what bases are good ?

C: How are the index set and the quantization to be chosen ?

D: Are there objects for which the approximation in bases is suboptimal ?

Historically, question A has been addressed by the information theory community in the context of rate-distortion theory. Shannon posed the problem in his 1948 landmark paper [52] and proved rate-distortion results in his 1959 paper [53]. Yet, as we have seen, rate-distortion theory has been mostly concerned with exact results within an asymptotic framework (the so-called large blocksize assumption together with random coding arguments). Thus, only particular processes (e.g. jointly Gaussian processes) are amenable to this exact analysis. The framework has however been used extensively, in particular in its operational version (when practical schemes are involved) [35].

In the stationary jointly Gaussian case the second question has a simple answer, based on rate-distortion theory. For any process, the Karhunen-Loève basis leads to the best linear approximation, due to its decorrelating properties. In the jointly Gaussian case the best possible approximation indeed happens to be a linear approximation, since decorrelation implies statistical independence. Yet, not all things in life are jointly Gaussian, and more powerful techniques than linear approximation can achieve a better rate-distortion tradeoff when the rate is constrained; that is where wavelets come into play, in conjunction with more general nonlinear approximation strategies[1]. For processes which are piecewise smooth (e.g. images), the abrupt changes are well captured by wavelets, and the smooth or stationary parts are efficiently represented by coarse approximations using scaling functions. Both practical algorithms (e.g. the EZW algorithm of Shapiro [54]) and theoretical analyses [7, 28] have shown the power of approximation within a wavelet basis or via a search in large libraries of orthonormal bases, based for example on binary subband coding trees. This leads to wavelet packets [9] and rate-distortion optimal solutions [45].

The third question is more complex than it looks at first sight. If there was no cost associated to the description of the index set, then I should clearly be the set $\{n\}$ such that

$$|\langle \varphi_n, f \rangle|_{n \in I} \geq |\langle \varphi_m, f \rangle|_{m \notin I} \tag{3.6}$$

But when the rate for I is accounted for, it might be more efficient to use a fixed set I for a class of objects, which needs no explicit indexing. For example, in the jointly Gaussian case, the optimal procedure chooses a fixed set of Karhunen-Loève basis vectors (namely

[1]The notion of "best" basis becomes slightly tricky if we allow for signal-dependent bases (to which the KLT also belongs). Indeed, suppose we want to code the Mona Lisa image; then the best basis is clearly that in which the first vector is the Mona Lisa image itself: with this choice we need only one bit to code the data. Yet the coding gain is entirely offset by the cost of informing the decoder of the basis "structure". In fact, the choice of transform must be a compromise between sufficient generality and power of representation within a class of signals. Piecewise smooth functions are well approximated by wavelets with enough vanishing moments and by local polynomial expansions. These models are sufficiently general to apply to a wide variety of signals, even departing in some degree from the piecewise polynomial template. Other issues in the choice of transform include computational efficiency and rate-distortion behavior in the case of quantized, truncated representations. The comparisons in this chapter address these last two issues specifically.

those corresponding to the largest eigenvalues) and spends all the rate to describe the coefficients with respect to these vectors. Note that a fixed subset corresponds to a linear approximation procedure (before quantization, which is itself non-linear) while choosing a subset as in (3.6) is a non-linear approximation method. It is easy to come up with examples of objects for which non-linear approximation is far superior to linear approximation. Consider a step function on $[0,1]$, where the step location is uniformly distributed over the support interval. Take the Haar wavelet basis as an orthonormal basis for $[0,1]$. It can be verified that the approximation error using M terms is approximately

$$\epsilon_L \approx 1/M \tag{3.7}$$

for the linear case, while it is

$$\epsilon_{NL} \approx 2^{-M} \tag{3.8}$$

for a non-linear approximation using the M largest terms. However, this is only the first part of the rate-distortion story, since we still have to index the M chosen terms. Anticipating what will be proven in the following, we see that we have to represent a certain number of scales J and that, at each scale, the coefficients require a certain number of bits. This split leads to a number of scales $J \approx \sqrt{R}$. The error is the sum of errors of each scale, each of which is of the order $2^{-R/J}$. Together, we get:

$$D_{NL}(R) \approx \sqrt{R}2^{-\sqrt{R}} \tag{3.9}$$

Finally, the fourth question is a critical one. While it is very popular to use othogonal bases for approximation, this cannot be the end of the story. Just as not every stochastic process is Gaussian, not all objects will be well represented in an orthogonal basis. In other words, fitting a linear subspace to arbitrary objects is not always a good approximation. Furthermore, even for objects where a basis does well, some other approximation method might do much better. In the above step function example, for instance, a simple minded coding of the step location and of the step values leads to a rate-distortion behavior

$$D'(R) \approx 2^{-R/2} \tag{3.10}$$

The purpose of the following sections is to analyze these rate-distortion trade-offs more in detail, both from a theoretical and a practical point of view.

3.2 R/D upper bounds for piecewise polynomial functions

Consider a continuous time signal $s(t)$, $t \in [a,b]$, composed of M polynomial pieces; assume that the maximum degree of any polynomial piece is less than or equal to G and that each

piece (and therefore the entire signal) is bounded in magnitude by some constant A. The signal is uniquely determined by the M polynomials and by $M-1$ internal breakpoints; by augmenting the set of breakpoints with the interval extremes a and b, we can write:

$$s(t) = \sum_{n=0}^{G} p_n^{(i)} t^n = p_i(t) \quad \text{for } t_i \leq t < t_{i+1} \tag{3.11}$$

where $a = t_0 < t_1 < \ldots < t_{M-1} < t_M = b$ are the breakpoints and the $p_n^{(i)}$ are the i-th polynomial coefficients (with $p_n^{(i)} = 0$ for n larger than the polynomial degree); let $T = (b-a)$. We will consider two approximation strategies for this type of function; the first is based on an oracle providing with arbitrary accuracy the function parameters, and it encodes the separate pieces by means of a quantized representation via local Legendre polynomials. The second approximation method uses compact-support wavelets with $G+1$ vanishing moments and encodes the nonzero wavelet coefficients along with a significance map. For both methods, we will derive an (hopefully tight) upper bound on the R/D performance, and we will compare it to actual experimental results.

3.2.1 Oracle-based local modeling

As we said, assume that the values for M, for the degrees of the polynomial pieces, and for the internal breakpoints are provided with arbitrary accuracy by an oracle. In this case, the derivation of the operational R/D bound will be carried out in three steps: first we will determine a general R/D upper bound for the single polynomial pieces; secondly, we will determine an R/D upper bound for encoding the breakpoint values; finally, we will determine the jointly optimal bit allocation for the whole signal. Here and in the next section, we will always consider the rate large enough to legitimate representing the distortion as a function of a continuous parameter; in other words, we will assume that the high-resolution hypothesis holds for all quantizers.

Encoding of one polynomial piece

Consider the i-th polynomial of degree G_i, defined over the support $I_i = [t_i, t_{i+1}]$ of width S_i. Using a local Legendre expansion (see Appendix 3.A) we can write (the subscript i is dropped for clarity throughout this section):

$$p(t) = \sum_{n=0}^{G} p_n t^n = \sum_{n=0}^{G} \frac{2n+1}{S} l_n L_I(n;t) \tag{3.12}$$

where $L_I(n;t)$ is the n-th degree Legendre polynomial over I; due to the properties of the expansion it is

$$|l_n| \leq AS \tag{3.13}$$

for all n. The squared error after quantizing the coefficients can be expressed as

$$\begin{aligned} e^2 &= \sum_{n=0}^{G} \left(\frac{2n+1}{S}\right)^2 (l_n - \hat{l}_n)^2 \int_{t_i}^{t_{i+1}} L_I^2(n;t)\, dt = \\ &= S^{-1} \sum_{n=0}^{G} (2n+1)(l_n - \hat{l}_n)^2 \end{aligned} \tag{3.14}$$

where $\hat{l}_n$ are the quantized values. Assume using for each coefficient a different b_n-bit uniform quantizer over the range specified by (3.13) for a step size of $2AS2^{-b_n}$; the total squared error can be upper bounded as:

$$e^2 \leq D_p = A^2 S \sum_{n=0}^{G} (2n+1) 2^{-2b_n} \tag{3.15}$$

For a global bit budget of R_p bits, with R_p sufficiently large, the optimal allocation can be found by solving the reverse waterfilling problem

$$\begin{cases} \dfrac{\partial D_p}{\partial b_n} = \text{constant} \\ \sum b_n = R_p \end{cases} \tag{3.16}$$

which yields

$$b_n = \frac{R_p}{G+1} + \log_2 \sqrt{\frac{2n+1}{\bar{C}}} \tag{3.17}$$

with

$$\bar{C} = \left[\prod_{n=0}^{G} (2n+1)\right]^{\frac{1}{G+1}}; \tag{3.18}$$

since the geometric mean is always less than or equal to the arithmetic mean we have $\bar{C} \leq (G+1)$, and we finally obtain the following upper bound for the i-th polynomial piece:

$$D_p(R_p) \leq A^2 S (G+1)^2 2^{-\frac{2}{G+1} R_p}. \tag{3.19}$$

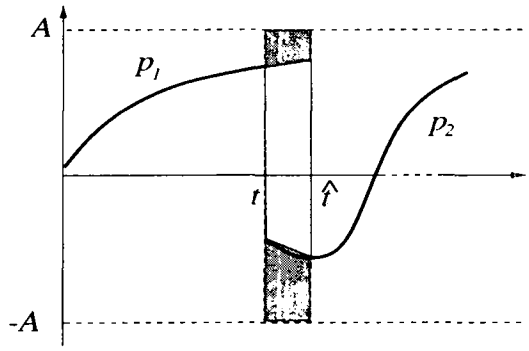


Figure 3.1: Encoding of switchpoints: true error (light area) and general upper bound (dark area)

Encoding of switchpoints

Assume that the $M+1$ switchpoints t_i, as provided by the oracle, are quantized with a uniform quantizer over the entire support of the signal. In terms of the overall mean squared error, the error relative to each quantized switchpoint can be upper bounded by (see Figure 3.1):

$$\epsilon_{t_i}^2 \leq 4A^2 \, |t_i - \hat{t}_i|. \tag{3.20}$$

Again, the magnitude of the error is at most one half of the quantizer's step size, so that for a given switchpoint we have:

$$D_t(R_t) \leq 2A^2 T \, 2^{-R_t} \tag{3.21}$$

where R_t is the quantizer's rate.

Composite R/D bound

The global distortion bound for $s(t)$ is obtained additively as

$$D \leq \sum_{i=1}^{M} D_{p_i}(R_{p_i}) + \sum_{i=0}^{M+1} D_{t_i}(R_{t_i}) \tag{3.22}$$

where $D_{p_i}(R_{p_i})$ and $D_{t_i}(R_{t_i})$ are the bounds in (3.19) and (3.21) respectively, and where the subscript denotes the index of the polynomial pieces.

In order to obtain the optimal bit allocation for the composite polynomial function given an overall rate, it would be necessary to find the constant-slope operating points for all the summation terms in (3.22), as shown in (3.16); the resulting formulas, however, would be entirely impractical due to their dependence on all the polynomial parameters across the whole function. Instead, we choose to derive a coarser but general upper bound by introducing the following simplifications:

- all polynomial pieces are assumed of maximum degree G; this implies that, for polynomials of lower degree, bits are allocated to the zero coefficients as well;
- the support of each polynomial piece S_i is "approximated" by T, the entire function's support; together with the previous assumption, this means that the waterfilling algorithm will assign the same number of bits R_p to each polynomial piece;
- the origin of the function support (a) is either known or irrelevant; this reduces the number of encoded switchpoints to M;
- all switchpoints are encoded at the same rate R_t.

With these simplifications the rate distortion bound becomes:

$$D(R) \leq A^2 T M \left(2^{-R_t+1} + (G+1)^2 2^{\frac{2}{G+1}R_p}\right) \tag{3.23}$$

where the total bit rate is $R = M(R_t + R_p)$. By the usual reverse waterfilling argument we obtain the optimal allocation:

$$R_p = \frac{G+1}{G+3}\frac{R}{M} + \log_2 K \tag{3.24}$$

$$R_t = \frac{2}{G+3}\frac{R}{M} - \log_2 K \tag{3.25}$$

with $K = (2G+2)^{(G+1)/(G+3)}$. Using the relation (for $G > 0$)

$$2K + (G+1)^2 K^{-\frac{2}{G+1}} \leq 2(G+1)^2 \tag{3.26}$$

a simplified global upper bound is finally:

$$D_P(R) \leq 2A^2 T M (G+1)^2 2^{-\frac{2}{G+3}\frac{R}{M}}. \tag{3.27}$$

3.2.2 Wavelet-based approximation

Consider now an orthonormal wavelet basis over $[a, b]$ where the wavelet has at least $G+1$ vanishing moments [8]; we will consider a quantized nonlinear approximation of $s(t)$ using a basis expansion over $[a, b]$, and bound the R/D behavior of the approximation following the lines in [7].

Distortion

If the wavelet has $G+1$ vanishing moments, then the only nonzero coefficients in the expansion correspond to wavelets straddling one or more switchpoints; since the wavelet has a compact support as well, each switchpoint affects only a finite number of wavelets at each scale, which is equal to the length of the support itself. For $G+1$ vanishing moments, the wavelet support L is

$$L \geq 2G + 1 \tag{3.28}$$

and therefore, at each scale j in the decomposition the number of nonzero coefficients C_j is bounded as

$$L \leq C_j \leq ML. \tag{3.29}$$

For a decomposition over a total of J levels, if we neglect the overlaps at each scale corresponding to wavelets straddling more than a single switchpoint we can upper bound the total number of nonzero coefficients C as

$$C \leq MLJ. \tag{3.30}$$

It can be shown [27] that the nonzero coefficients decay with increasing scale as

$$|c_{j,k}| \leq ATW\, 2^{-j/2} \tag{3.31}$$

where W is the maximum of the wavelet's modulus. Using the same high-resolution b-bit uniform quantizer for all the coefficients[2] with a stepsize of $2ATW\, 2^{-b}$ we obtain the largest scale before all successive coefficients are quantized to zero:

$$J = 2b - 2. \tag{3.32}$$

With this allocation choice the total distortion bound is $D = D_q + D_t$ where

$$D_q = \sum_k \sum_{j=0}^{J} (c_{j,k} - \hat{c}_{j,k})^2 \tag{3.33}$$

is the quantization error for the coded nonzero coefficients and where

$$D_t = \sum_k \sum_{j=J+1}^{+\infty} c_{j,k}^2 \tag{3.34}$$

[2] In [7] the authors carry out a more detailed analysis in which the quantizer's stepsize is varied according to the decay in (3.31); this however affects only the constants in the R/D bound and not the asymptotics.

is the error due to the wavelet series truncation after scale J (in both summations the index k runs over the nonzero coefficients in each scale). Upper bounding the quantization error in the usual way and using the bound in (3.31) for each discarded coefficient we obtain

$$\begin{aligned} D &\leq C(ATW)^2 2^{-2b} + ML(ATW)^2 2^{-J} = \\ &= ML(ATW)^2 \left(1 + \frac{1}{4}J\right) 2^{-J} \end{aligned} \tag{3.35}$$

Rate

Along with the quantized nonzero coefficients, we must supply a significance map indicating their position; due to the structure of $s(t)$, 2 bits per coefficients suffice to indicate which of the next-scale wavelet siblings (left, right, both, or none) are nonzero. The total rate therefore is

$$R = C(b+2) \leq MLJ(J/2+3) \tag{3.36}$$

where we have used (3.30) and (3.32). In our high-rate hypothesis it is surely going to be $b \geq 4$ and therefore we can approximate (3.36) as

$$R \leq MLJ^2. \tag{3.37}$$

Global upper bound

Eqn. (3.35) provides a distortion bound as a function of J; in turn, J is a function of the overall rate as in (3.37). Combining the results we obtain the overall rate/distortion bound:

$$D_W(R) \leq (ATW)^2 M(2G+1) \left(1 + \frac{1}{4}\sqrt{\frac{1}{2G+1}\frac{R}{M}}\right) 2^{-\sqrt{\frac{1}{2G+1}\frac{R}{M}}} \tag{3.38}$$

where we have assumed a minimum support wavelet, for which $L = 2G+1$.

3.2.3 Comments and experimental results

To recapitulate, the two upper bounds obtained in the previous sections are of the form:

$$\begin{aligned} &\textit{polynomial approximation:} && D_P(R) = C'_p 2^{-C_p R} \\ &\textit{wavelet approximation:} && D_W(R) = C'_w(1 + \alpha\sqrt{C_w R})2^{-\sqrt{C_w R}} \end{aligned}$$

Since these are upper bounds, we are especially concerned with their tightness. Unfortunately, as we have seen, many simplifications have been introduced in the derivation,

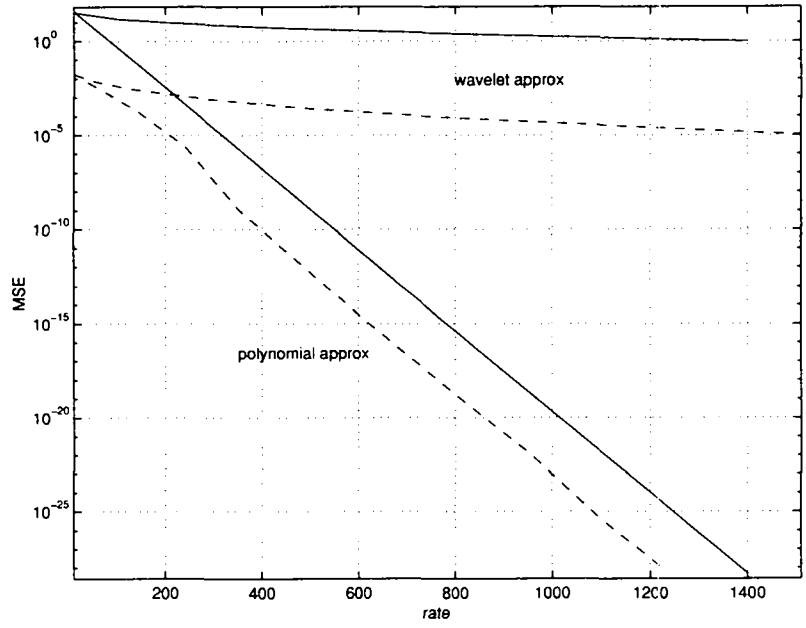


Figure 3.2: Theoretical (solid) and experimental (dashed) R/D curves.

some of which are definitely rather crude; we will therefore concentrate on the rate of decay of the R/D function rather than on the exact values of the constants. In order to gauge the applicability of the theoretical bounds, we can try to compare them to the actual performance of practical coding systems. A word of caution is however necessary: in order to implement the coding schemes described above, which are derived for continuous time functions, a discretization of the test data is necessary; as a consequence, an implicit granularity of the time axis is introduced, which limits the allowable range for both the breakpoint quantization rate and for the number of decomposition levels in the wavelet transformation. Unfortunately, computational requirements soon limit the resolution of the discretization: in our experiments we have used 2^{16} points. The two approximation techniques have been applied to randomly generated piecewise polynomial functions with parameters $A = 1$, $T = 1$, $G = 4$ and $M = 4$; Daubechies wavelets with 5 vanishing moments on the $[0, 1]$ interval have been used for the decomposition. The results are shown in Figure 3.2: the solid lines and the dashed lines display the R/D bound and the operational R/D curve respectively for the polynomial and wavelet approximation strategies averaged over 50 function realizations.

A closer inspection of the R/D curves shows that, especially for the wavelet case, there appears to be a large numerical offset between theoretical and practical values even

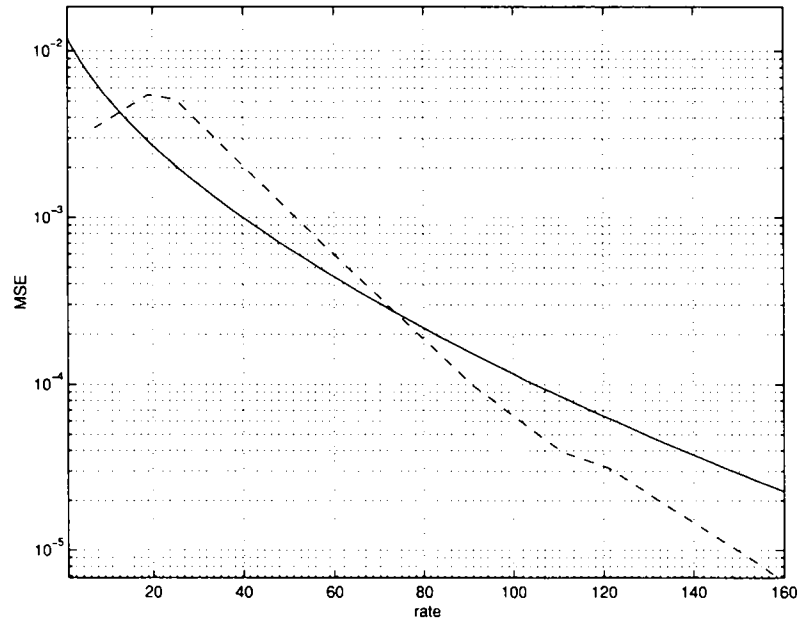


Figure 3.3: Theoretical (solid) and experimental (dashed) R/D curve for the Haar wavelet approximation of the step function.

though the rate of decay is correct. This simply indicates that the bounds for the constants in (3.38) are exceedingly large and the question is whether we can arrive at tighter estimates. In the absence of a detailed statistical description for the characteristic parameters of $s(t)$, the answer remains rather elusive in the general case; we can however try to develop our intuition by studying in more detail a toy problem involving minimal-complexity elements and a simple statistical model for the approximated function. The availability of a particular statistical model for the generating process allows us to derive an R/D result in expectation, which is hopefully tighter.

Consider the simple case in which $G = 0$, $M = 2$, $T = 1$, and $A = 1/2$: the resulting $s(t)$ is simply a step function over, say, $[0, 1)$; we will assume that the location of the step transition t_0 is uniformly distributed over the support and that the values of the function left and right of the discontinuity are uniformly distributed over $[-1/2, 1/2]$. Having a piecewise constant function allows us to use a Haar wavelet decomposition over the $[0, 1]$ interval; recall that the Haar scaling function and wavelet have a single vanishing moment

and they admit the closed-form representation

$$\varphi_0(t) = 1 \tag{3.39}$$

$$\psi_{j,k}(t) = \begin{cases} +2^{j/2} & k2^{-j} \le t < (k+1/2)2^{-j} \\ -2^{j/2} & (k+1/2)2^{-j} \le t < (k+1)2^{-j} \\ 0 & \text{elsewhere} \end{cases} \tag{3.40}$$

from which it is easy to see that there is no overlap between wavelets within a scale. Now the following facts hold:

- because of the absence of overlap, at each scale we have exactly one nonzero coefficient[3]; the relation in (3.30) becomes exact:

$$C = J; \tag{3.41}$$

- under the high resolution hypothesis for a b-bit quantizer, the quantization error becomes a uniform random variable over an interval half a step wide; the expected error for each quantized coefficient is therefore $2^{-2b}/12$;

- the series truncation error (3.34) is, in expectation,

$$E[D_t] = (1/36)\, 2^{-(J+1)}; \tag{3.42}$$

  (for a proof, see Appendix 3.B);

- again, due to the non overlapping properties of the Haar wavelet, we can rewrite (3.37) simply as $R \le J^2$.

With these values, the R/D bound, *in expectation*, becomes:

$$D_W(R) \le \frac{1}{72}\left(1 + \frac{3}{4}\sqrt{R}\right) 2^{-\sqrt{R}}. \tag{3.43}$$

Figure 3.3 displays this curve along with the experimental results (dashed line); we can now see that the numerical values agree to within the same order of magnitude. For completeness, the expected R/D bound for the polynomial approximation of the above step function (obtained with a similar, simplified analysis) turns out to be:

$$D_P(R) = \frac{1}{6\sqrt{2}}\, 2^{-R/2} \tag{3.44}$$

and this curve, together with its experimental counterpart, is displayed in Figure 3.4.

[3]In fact, a further consequence of the non-overlapping wavelets is that we need only one bit per coefficient to encode the significance map; for simplicity we will however use the general rate requirement (3.36) both in the theoretical derivation and in the algorithmic implementation.

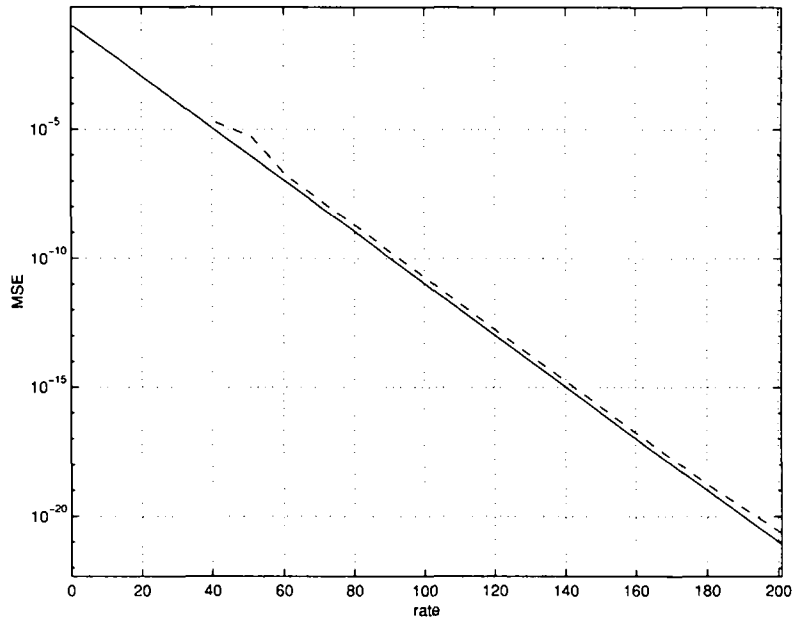


Figure 3.4: Theoretical (solid) and experimental (dashed) R/D curve for the polynomial approximation of the step function.

3.3 R/D optimal approximation

We have seen that the direct polynomial approximation displays a far better rate-distortion asymptotic behavior than the standard nonlinear wavelet approximation. However, the polynomial bound was derived under two special hypotheses which are not generally met in practice: the availability of an oracle and the use of high resolution quantizers. Since the goal of most approximation techniques is a parsimonious representation of the data for compression purposes, the question arises naturally: what is the best coding strategy in a practical setting where the polynomial parameters are initially unknown and the bit rate is severely constrained? If by "best" we mean the optimal solution with respect to an additive distortion measure, then the answer is given by the dynamic segmentation/allocation algorithm introduced in the previous chapter. As we have seen, in this constrained allocation approach, both breakpoints and local models must be determined jointly, since the optimal segmentation is a function of the family of approximants we allow for and of the global bitrate. In particular, for low rates, the available resources might not allow

for a faithful encoding of all pieces and a globally optimal compromise solution must be sought for, possibly by lumping several contiguos pieces into a single one, or by approximating some pieces by lower-degree polynomials with lighter description complexity. We will see that the algorithm performance matches, and extends to the low bitrate case, that of the oracle-based modeling. Please note that now we are entering an algorithmic scenario where we perforce deal with discrete-time data vectors rather than continuous-time functions; similarly to the experimental results of the previous section, granularity of the involved quantities and computational requirements are now important factors.

3.3.1 Operational setup

Consider an N-point data vector $\boldsymbol{x} = x_1^N$, which is a sampled version of a piecewise polynomial function $s(t)$ over a given support interval. The family of approximation models we choose to use is the crucial "engineering" decision of the problem and is ruled by a-priori knowledge on the input data (polynomial pieces of maximum degree G) and by economical considerations in terms of computational requirements. In particular, we choose a fixed, limited set of possible rates associated to a polynomial model of given degree, with quantization of the individual coefficients following the line of (3.17). Clearly, the validity of such design parameters can only be assessed via the performance measure yielded by the operational R/D curve. In the following we will assume a family of K polynomial models, which is the aggregate set of polynomial prototypes from degree 0 to G with different quantization schemes for the parameters.

For a given segmentation $\boldsymbol{t}$ and a related allocation $\boldsymbol{w}$, defined as in section 2.2.4, $R(\boldsymbol{t}, \boldsymbol{w})$ is the cost, in bits, associated to the sequence of $\sigma(\boldsymbol{t})$ polynomial models and $D(\boldsymbol{t}, \boldsymbol{w})$ is the cumulative squared error of the approximation. The polynomial coefficients are estimated by solving a Least Squares (LS) problem over each segment for all orders; the resulting coefficients are then quantized according to one of the possible quantization schemes we allow for. We can therefore write

$$D(\boldsymbol{t}, \boldsymbol{w}) = \sum_{i=1}^{\sigma(\boldsymbol{t})} d(t_i, t_{i+1}; \hat{\boldsymbol{p}}(w_i)); \tag{3.45}$$

where

$$d(t_i, t_{i+1}; \hat{\boldsymbol{p}}(w_i)) = \| V\, \hat{\boldsymbol{p}}(w_i) - x_{t_i}^{t_{i+1}-1} \|^2; \tag{3.46}$$

here, V is a suitably defined Vandermonde matrix (see later for details) and $\hat{\boldsymbol{p}}(w_i)$ is a vector containing the estimated and quantized polynomial coefficients for the i-th segment according to model w_i (w_i indicates both the polynomial order and the quantization scheme). We will assume that the polynomial coefficients are coded independently and

therefore their overall cost (in bits) is a function $b(\cdot)$ of the model's index only. A constant part of this cost is used up by the model index itself and by the segment's length.

3.3.2 Implementation

Building Blocks

In the implementation of the dynamic segmentation algorithm we have chosen a simplified set of quantization schemes. At low bitrates, Equation (3.17) states that the optimal bit distribution for a set of polynomial coefficients is basically uniform. We choose four possible allocations of 4, 8, 12, and 16 bits for the single coefficient, so that the total rate of a single polynomial piece is linearly dependent on its degree; the family of models has thus cardinality $K = 4(G+1)$. To each polynomial piece is associated a side information block comprising $\lceil \log_2 K \rceil$ bits for the model index and $\lceil \log_2 N \rceil$ bits for the segments length.

Incremental Least-Squares Solution

In order to find the optimal segmentation the dynamic programming Equation (2.20) requires us to solve a least squares problem for all segments $x_n^m, 1 \leq n \leq N, n \leq m \leq N$; this can be efficiently carried out in an incremental fashion, and the trellis structure in Algorithm 2.2 will be of help in organizing all the quantities involved in the computation. For a data segment x_m^n of length $L = n - m + 1$ the minimum squared-error polynomial approximation of order D is found by solving the LS problem:

$$\min_{\boldsymbol{p}} \|V_{L,D}\,\boldsymbol{p} - x_n^m\|^2 \tag{3.47}$$

(all vectors are colums vectors) where the $\boldsymbol{p} = p_0^D$ is a vector of $D+1$ polynomial coefficients and $V_{L,D}$ is the following $L \times (D+1)$ Vandermonde matrix:

$$V_{(L,D)} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 2 & 2 & 2^2 & \dots & 2^D \\ & & \dots & & \\ 1 & L & L^2 & \dots & L^D \end{bmatrix} \tag{3.48}$$

The solution to the LS problem is achieved by means of a QR factorization: assume we have already computed $V_{L,D} = QR$, with $Q \in \mathbb{R}^{L \times L}$ an orthogonal matrix and $R \in \mathbb{R}^{L \times (D+1)}$ upper triangular, so that its last $L - D - 1$ rows are identically zero. Then the minimizing $\boldsymbol{p}$ satisfies

$$\begin{bmatrix} R_0 \\ R_1 \\ \dots \\ R_D \end{bmatrix} \boldsymbol{p} = \begin{bmatrix} Q_0^t \\ Q_1^t \\ \dots \\ Q_D^t \end{bmatrix} x_n^m \tag{3.49}$$

where R_i is the i-th row of R and Q_i^t is the i-th row of the transpose of Q. This system of equations can be solved in $O(LD)$ time. Now, if we extend the segment with a new data point to x_n^{m+1}, we have to solve a related problem in which a new row has been appended to the previous Vandermonde matrix, extending it to $V_{L+1,D}$. Luckily, the QR factortization need not be recomputed from scratch, but can be extended by means of Givens rotations [18], and the new solution to (3.49) can be found in $O(LD)$ time as well. Therefore, for a polynomial fitting of any degree D, the initial QR factorization of $V_{D+1,D}$ can be precomputed and stored in memory, and the LS problems for all segment of length from $D+1$ to N can be computed incrementally in $O(N^2)$ time. The approximation error is then computed by quantizing the coefficients to $\hat{\boldsymbol{p}}$ and explicitly evaluating $\|V\hat{\boldsymbol{p}} - \boldsymbol{x}\|^2$; this requires once again $O(LD)$ computations per length-L segment, for a total of $O(N^2)$ operations.

Trellis Algorithm

The trellis algorithm for the dependent allocation case can be adapted to the problem at hand in the following way:

Algorithm 3.1: Local Polynomial Modeling

Assume we have already pre-computed the QR factorization of the Vandermonde matrices $V_{i,i}$ for $i = 0, \ldots, G$.

Step 1: Initialization

1 Build the trellis: for $1 \leq n \leq N$
create a sequence of sets S_n containing n new states $s_{n,m}$ each, $1 \leq m \leq n$.

2 For $i = 0, \ldots, G$
For $n = 1, \ldots, N$
For $m = n, \ldots, N$
Extend the QR factors of $V_{n-1,i}$ to the QR factors for $V_{n,i}$;
Compute the distortion and rate values $d(n - m + 1, n; k)$, $r(k)$ for $1 \leq k \leq K$ and associate these values to $s_{n,m}$.

From this point on one can proceed as in Algorithm 2.2.

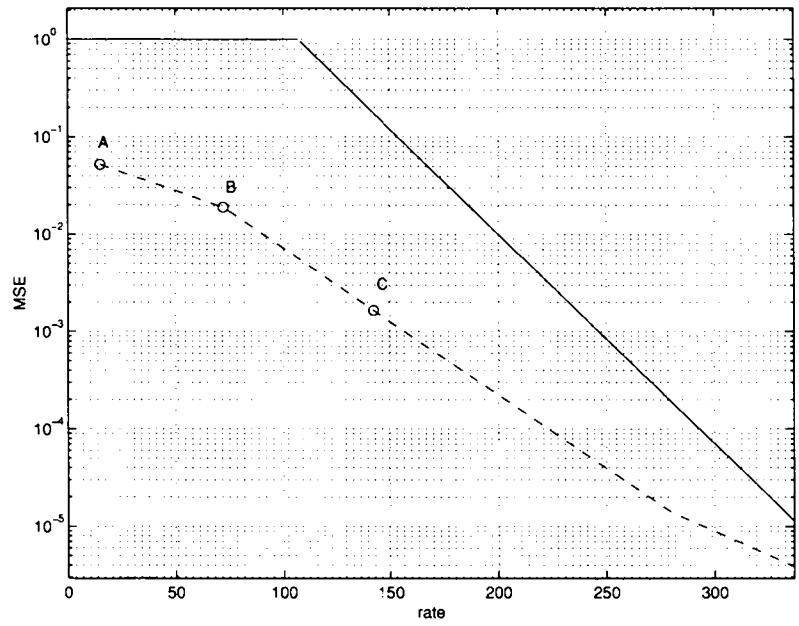


Figure 3.5: Theoretical (solid) and experimental (dashed) R/D curves for the dynamic segmentation algorithm.

3.3.3 Results

We can now compare the experimental results of the optimal allocation algorithm with the polynomial approximation R/D bound obtained using an oracle; however, since here the interest lies in very low bit rates as well, we need to somehow refine the bound in (3.27). In fact, under severe rate constraints, there might not be enough bits to encode the exact structure of the function and the dynamic algorithm will be forced to use a coarse segmentation in which several contiguous polynomial pieces are approximated by just one model; in the limit, when the rate goes to zero, the approximation error approaches the integral of $s^2(t)$ over the entire support of the function. This is not reflected by Equation (3.27), where the expression for the error always assumes M distinct pieces. By approximating the maximum error by $4A^2T$, we can define a more appropriate R/D bound for the poly-

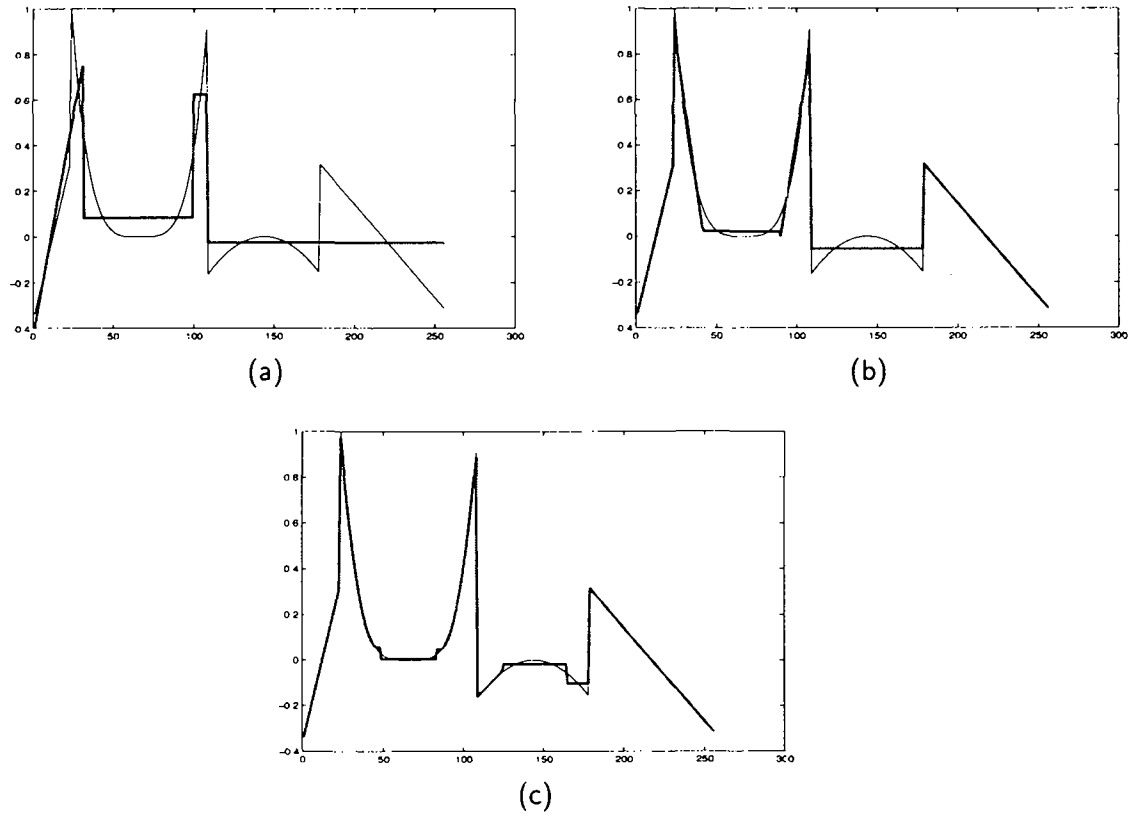


Figure 3.6: Approximations provided by the dynamic segmentation algorithm corresponding to points A, B, and C on the R/D curve in Figure 3.5.

nomial case as

$$D'_P(R) = \min\{4A^2T, D_P(R)\} \tag{3.50}$$

Figure 3.5 shows the numerical results obtained for a set of piecewise polynomial functions as in the previous experiment; the underlying sampling is however coarser here ($K = 2^8$), due to the heavier computational load. As before, the solid line indicates the new theoretical upper bound and the dashed line the R/D performance of the dynamic algorithm. It is also interesting to look more in detail at the segmentation/allocation choices performed by the algorithm for different bitrate constraints; this is displayed in Figures 3.6-(a),(b),

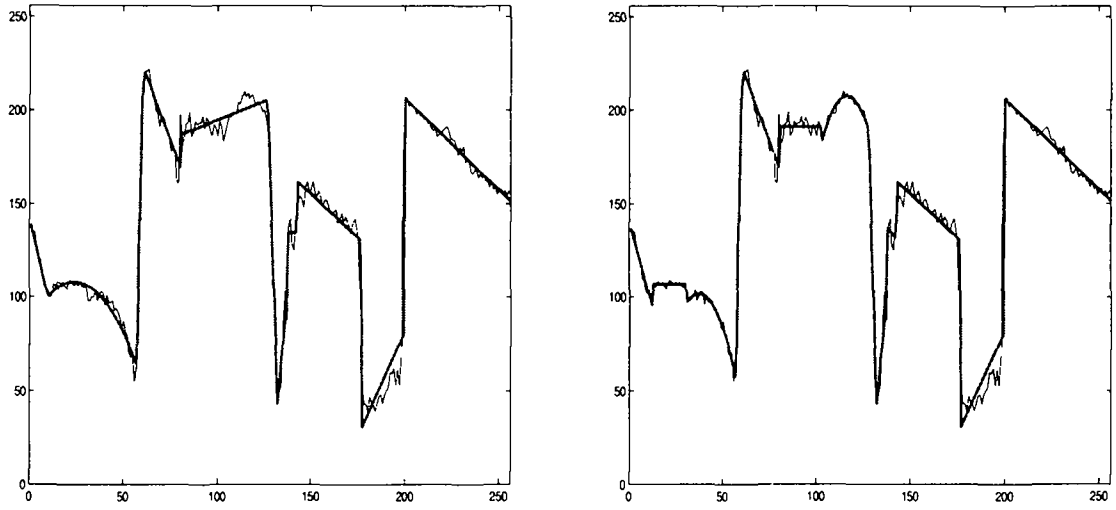


Figure 3.7: Piecewise polynomial approximations of a line of Lena.

and (c) with respect to the R/D points A, B, and C marked by a circle in Figure 3.5. In Figure 3.6 the thin line shows the original piecewise polynomial function while the thick lines shows the algorithmic results; while not always very intuitive, these low bit rate approximations are nonetheless optimal in a MSE sense.

3.3.4 Whereto tends all this?[4]

As a final note, we can ask ourselves two further questions: how does this framework extend to real-world signals, which are clearly not exactly piecewise polynomial ? And more, can this framework be applied and compared to practical coding scenarios in which wavelets are known to perform very well, such as image compression ? Unfortunately, dynamic programming techniques do not work for two-dimensional problems, and it is not clear how to fit polynomial surfaces in a globally optimal way. Yet, we can gain some intuition about both questions by looking at Figures 3.7-(a) and (b). The thin line represents a single column of the "Lena" image, for a total of 256 pixels; the thick lines are the piecewise polynomial approximations of the data, at increasing rates, obtained with the dynamic segmentation algorithm introduced above. We could argue that, for increasing bitrates, the algorithm captures more and more finely the local polynomial trends under-

[4] -*W. Shakespeare.*

(a)

(b)

Figure 3.8: Original (a) and columnwise approximation (b) of the Lena image.

lying the image surfaces, while the finer details can be represented as an additive, noise-like residual; this mirrors a similar type of signal decomposition which has been successfully used in the field of audio analysis [51]. A simple-minded demonstration of the general idea can be obtained by running the polynomial modeling algorithm columnwise over the entire image; the results are shown in Figure 3.8-(b) (Figure 3.8-(a) being the original). The coding algorithm neglects horizontal correlations and therefore the effect is that of a "rain-streaked" picture; whether improvements may lead to an efficient approximation scheme for real images is however hard to say at present. All this represents an interesting line of research for future extensions of this work.

3.4 Summary

This chapter made use of the dynamic segmentation/allocation framework to validate a rate-distortion bound for the coding performance of piecewise polynomial functions. Section 1 introduced the problem of data compression based on basis expansion, with a particular attention to wavelet coding methods and to nonlinear approximations techniques. In Section 2 the first R/D bound was derived for a local polynomial expansion based on an oracle, for a behavior of the form of $R(D) \approx 2^{-R}$; in Section 3 the same approach was applied to a nonlinear wavelet coding method with uniform quantization, for an R/D bound of the form of $R(D) \approx \sqrt{R}2^{-\sqrt{R}}$. Section 3 introduced the dynamic framework, which obtains results very similar to the oracle-based method; algorithmic details were discussed and experimental results illustrated.

Appendix 3.A Local Legendre Expansion

Legendre polynomials are usually defined over the $[-1,1]$ interval by the recurrence relation

$$(n+1)L(n+1;t) = (2n+1)t\,L(n;t) - nL(n-1;t) \tag{3.51}$$

where $L(n;t)$ is the Legendre polynomial of degree n. They constitute an orthogonal basis for $L[-1,1]$:

$$\int_{-1}^{1} L(n;t)L(m;t)\,dt = \frac{2}{2n+1}\delta(n-m) \tag{3.52}$$

and in particular, for a polynomial $p(t)$ of degree G over $[-1,1]$, we can write

$$l_n = \int_{-1}^{1} L(n;t)p(t)\,dt, \quad n = 0,\ldots,G \tag{3.53}$$

$$p(t) = \sum_{n=0}^{G} \frac{2n+1}{2} l_n L(n;t). \tag{3.54}$$

Since $|L(n;t)| \leq 1$ for all n, $|l_n| \leq 2\sup_{[-1,1]}[p(t)]$.

A *local* Legendre expansion over the interval $I = [\alpha,\beta]$ can be obtained by defining a translated set of orthogonal polynomials

$$L_I(n;t) = L(n; \frac{2}{\beta-\alpha}t - \frac{\alpha+\beta}{\beta\alpha}); \tag{3.55}$$

the orthogonality relation becomes

$$\int_{\alpha}^{\beta} L_I(n;t)L_I(m;t)\,dt = \frac{\beta-\alpha}{2n+1}\delta(n-m) \tag{3.56}$$

and the analysis/synthesis formulas can be written as:

$$l_n = \int_{\alpha}^{\beta} L_I(n;t)p(t)\,dt, \quad n = 0,\ldots,G \tag{3.57}$$

$$p(t) = \sum_{n=0}^{G} \frac{2n+1}{\beta-\alpha} l_n L_I(n;t). \tag{3.58}$$

Appendix 3.B Estimate of the series truncation error

The estimate in (3.42) can be obtained as follows: assume that at scale j_0 the step discontinuity at t_0 falls within the interval $[h2^{-j_0}, (h+1)2^{-j_0})$ for some h. Then in the Haar wavelet series for $s(t)$ the indices of the nonzero coefficients $c_{j,k}$ for $j > j_0$ satisfy

$$h2^{j-j_0} \leq k < (h+1)2^{j-j_0}. \tag{3.59}$$

The wavelet set $\{\psi_{j,k}(t)\}$ with j and k as above form an orthonormal basis for the $[h2^{-j_0}, (h+1)2^{-j_0})$ interval minus the addition of a scaling function $\varphi(t) = 2^{j_0/2}$ over the same interval. We can therefore write:

$$\begin{aligned} D_t &= \sum_{j=0}^{\infty} \sum_{k=h2^{j-j_0}}^{(h+1)2^{j-j_0}-1} c_{j,k}^2 = \\ &= \int_{h2^{-j_0}}^{(h+1)2^{j-j_0}} s^2(t)\,dt - \left[2^{-j_0/2} \int_{h2^{-j_0}}^{(h+1)2^{j-j_0}} s(t)\,dt\right]^2 \end{aligned} \tag{3.60}$$

where we have used Parseval's identity.

Now consider the location of the step (see Figure 3.9); let $\tau = t_0 - h2^{-j_0}$ be the distance between the discontinuity and the origin of the interval. Due to the properties of $s(t)$ we can safely assume that τ is uniformly distributed over $[0, 2^{-j_0}]$. We can now write

$$D_t = \tau x_1^2 + (2^{-j_0} - \tau)x_2^2 - 2^{j_0}(\tau^2 x_1^2 + (2^{-j_0} - \tau)^2)x_2^2 + 2\tau(2^{-j_0} - \tau)x_1 x_2 \tag{3.61}$$

where $x_{1,2}$ are the values of $s(t)$ left and right of the jump, respectively. Taking expectations over the independent quantities τ, x_1, and x_2, where $x_{1,2} \in \mathcal{U}[-1/2, 1/2]$, we finally have:

$$E[D_t] = (1/36)\,2^{-j_0}. \tag{3.62}$$

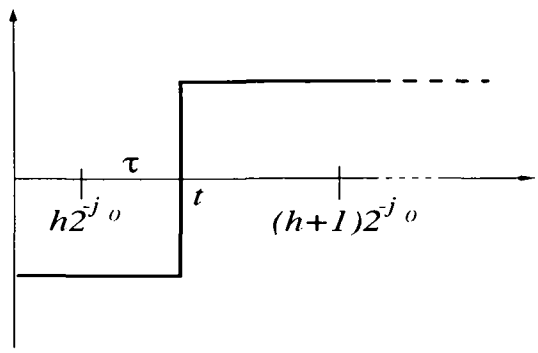


Figure 3.9: Switchpoints location at a given wavelet scale.

Chapter 4

Data Compression: Linear Prediction and Arithmetic Coding

Dum loquimur fugerit invida aetas ...
– Q. HORATIUS FLACCUS, *Carmina 1,11*

After the somehow "syntethic" case study of the previous chapter, we will now examine two practical applications of optimal segmentation techniques in the domain of data compression. In the first example we will treat the case of speech compression based on linear prediction. Speech is probably *the* piecewise stationary signal in the signal processing literature. Its characteristics are very well understood and even if most compression schemes are based on the fixed-window paradigm, so much study and fine tuning went into their design that their performance is usually remarkable. Linear prediction, in particular, is a spectacular example of how effective the "classic" random signal processing approach can be: time windowing to evoke stationarity, rational transfer function modeling, linear inverse filtering – it works beautifully, quickly and elegantly. Yet there's something to be gained even here; we will try to show that, if we really want to shave off the last bits from the LP representation of a speech signal, a dynamic segmentation approach is an effective way to go.

In the second application we will consider the case of arithmetic coding. Arithmetic coding is an extremely versatile source coding scheme which, when the exact input statistics are known, promises a compression performance to within two bits of the source en-

tropy. The problem obviously lies with estimating these statistics as accurately as possible, and a wide variety of adaptive coding schemes exists in the literature. Adaptivity can be of two kinds: in backward adaptive algorithms, only past data is used for the estimation, while in forward adaptive algorithms one is allowed to "peek ahead"; the latter case is in principle more accurate since it avoids the mismatches due to abrupt signal transitions, but it has the disadvantage that side information must be conveyed to the decoder. Whether a backward or a forward adaptive scheme is more advantageous depends obviously on the data realization. The idea here is to explore all data segmentations and test both schemes on each segment in order to minimize the size of the compressed data. Clearly, such an approach is suitable for piecewise stationary signals, for which we expect stable statitstics over disjoint data segments (forward coded) possibly linked by transitional regions with less identifiable trends (backward coded).

4.1 Speech compression via Linear Prediction

Arguably, linear prediction (LP) is amongst the most ubiquitous and successful signal processing tools, with applications ranging from channel equalization to data compression, and with a wealth of efficient and very robust algorithmic implementations. As a particular case of the more general least squares system identification problem, linear prediction relies on the special assumption that the unknown system is purely autoregressive; this is tantamount to postulating that the data are the output of an all-pole IIR linear filter $B(z) = 1/A(z)$. One can show that, under ideal conditions, linear prediction computes the exact inverse filter $A(z)$ together with the correct model order; inverse filtering the data by $A(z)$ recovers the original filter input [22].

In speech coding a whole class of coders, generally grouped under the label of *hybrid time-domain coders*, possesses a LP block at its core. This stems from the intuitively sound and physically relevant modelization of the human speech apparatus as a excitator-resonator pair: the oscillation of the vocal cords or a turbulent air flow are the input to the vocal tract in producing voiced and unvoiced sounds. Although the vocal tract is not exactly an all-pole IIR (there are zeros related to the nasal cavities), the accuracy of the achievable modelization is striking. After inverse filtering, the excitation (or *residual*) is recovered and different compression schemes are defined by the way the residual and the LP coefficients are encoded. At one extreme of simplicity, systems such as LPC-10 [58] and RELP [13] simply quantize the LP coefficients and a parametrized version of the residual in open-loop fashion. At the other extreme, codebook-based encoders such as CELP [49] utilize the LP direct filter to produce a speech waveform from a synthetic residual; the residual is constructed from codebook entries as to minimize (in some perceptual "norm") the error between original and resynthesized speech, with the encoder operating in closed-loop.

It is important to note that, although the closed-loop encoding process for a speech coder is usually optimized with respect to ad-hoc perceptual criteria, the inverse filter computed by the LP block is always obtained via a least squares minimization, where the goodness-of-fit is measured by the mean squared error (MSE). Here we are concerned precisely with the global optimization of the composite linear predictor for an arbitrary signal with respect to its overall squared error and its composite order. Indeed, the fundamental difficulty with linear prediction of real signals lies with their inherent nonstationarity; this is usually addressed by means of fixed-length windowing in the expectation that, over small time intervals, the signal varies little. While this is true in general terms, it does still happen that abrupt signal transitions fall within a windowed segment, and the probability of such events increases with the window size. On the other hand, the window size cannot be reduced arbitrarily since, for locally stationary segments, the accuracy of the estimation increases with it. Clearly, one would like to obtain a flexible segmentation in which the boundaries coincide with the transitions in the signal and the length of the segments minimize the local squared error. Such a flexible segmentation implies asynchronous coding and buffering; while this is bad news for low-delay applications, it opens the way to unequal resource allocation for different portions of the signal. It is a well known fact, for instance, that unvoiced speech can be coded with a coarser LP description than voiced speech, since it has a flatter spectral structure. For a given bit budget, LP parameters should receive more bits where they are most needed.

The segmentation/allocation techniques studied so far are indeed able to determine the optimal segmentation and the optimal sequence of predictors for the segments with respect to the global LP error. The approach differs from previous results attempting a local optimization of the LP encoding process [14, 16] in that we completely do away with frame-based encoding; the segmentation is completely flexible and the final cost (in bits) of the LP parameters is minimized from within the LP encoding process.

4.1.1 Linear Prediction

In the classic stochastic formulation, given a discrete time stationary signal $x(n)$, an order-p linear predictor computes p coefficients $a_1, \ldots, a_p$ so that the linear combination

$$\hat{x}(n) = a_1 x(n-1) + a_2 x(n-2) + \ldots + a_p x(n-p) \tag{4.1}$$

minimizes the expected squared error:

$$d^2 = \mathrm{E}\{|x(n) - \hat{x}(n)|^2\}. \tag{4.2}$$

Minimization of (4.2) with respect to the a_i's yields a set of *normal equations* which are of the form

$$\sum_{i=1}^{p} a_i \rho(i,j) = \rho(0,j), \text{ for } j = 1, \ldots, p \tag{4.3}$$

where $\rho(i,j) = \mathrm{E}_n\{x(n-i)x(n-j)\}$, the signal's autocorrelation at lag $|i-j|$.

For a finite data set, the problem loses its probabilistic nature and becomes a particular case of least squares approximation. In fact, for a truly autoregressive signal with no added noise, p samples suffice to uniquely determine the order-p generating model using Padè's method [25]; these hypotheses are of course never met in practice so that many more data points must be taken into account to minimize the effects of noise and model mismatch. For a data set x_m^{m+M-1}, $M \gg p$, the goal of linear prediction becomes the minimization of the squared error

$$d^2 = \sum_{n=A}^{B} (x_n - \hat{x}_n)^2; \tag{4.4}$$

the summation limits, A and B, reflect the implicit assumptions on the signal values outside of the known support and determine the influence of "border effects" on the LP computation. For $A = m + p$ and $B = m + M - 1$, the error is evaluated only within the interval, and no further assumptions are made; this LP strategy is called the *covariance method*. For $A = m$ and $B = m + M - 1 + p$, together with the assumption that the signal is zero-padded as necessary, we have the so-called *autocorrelation method*; this method often requires a tapering window since the discontinuity between the signal and the zero extension might have adverse effects on the parameter estimation, but it is the most advantageous computationally. In both cases, the deterministic LP problem produces a set of normal equations formally identical to (4.3) in which

$$\rho(i,j) = \sum_{n=A}^{B} x_{n-i} x_{n-j}. \tag{4.5}$$

The formulation in (4.5) is very close to the empiric autocorrelation formula; for an ergodic signal this would converge to the true autocorrelation as the sample size grows to infinity, and this formal analogy explains why it is customary to apply random processes terminology (such as "stationary") even to finite data sets.

The goodness-of-fit indicator for the deterministic linear predictor is the error in (4.4), usually normalized by M (the MSE); this measure is however blind to the specific causes of misadjustment and, for a fixed set of samples, it reacts similarly to phenomena such as additive noise, undermodeling, and signal transitions. However, if the data set is

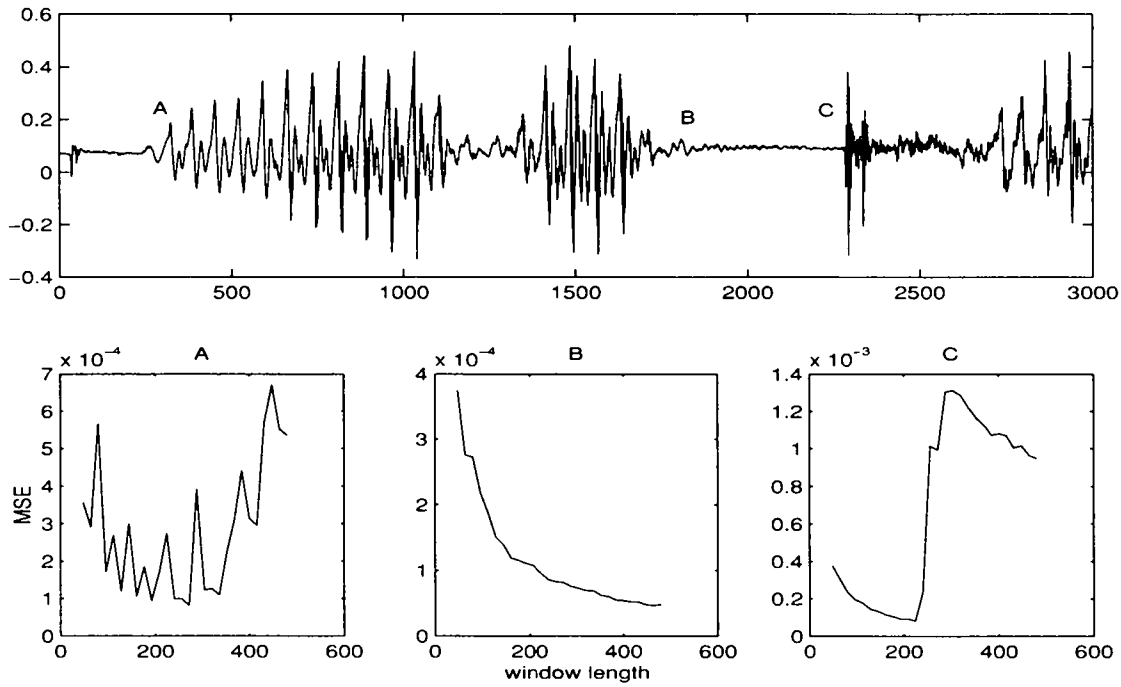


Figure 4.1: Speech segment (upper panel) and linear prediction MSE as a function of window length for data starting at points A, B, and C (three lower panels respectively).

a windowed portion of a longer signal, the behavior of the MSE as a function of window length and window placement offers more insight. Figure 4.1 shows a portion of a typical speech signal; the three lower panels show the MSE as a function of the window length for the three points in the signal marked A, B, and C. The first panel is relative to the onset of a voiced sound, and the MSE decreases non monotonically until the drift in the periodic waveform make it increase again; the non monotonic behavior in the the curve is due to the data window straddling, at times, a non integer number of periods. The second panel shows a monotonically decreasing MSE corresponding to a slightly noisy, silent gap between utterances. The third panel shows the abrupt transition in the MSE occurring at sharp signal onsets such as at point C. These simple examples suffice to demonstrate the potential benefits of a dynamic window size.

As for the order of the linear predictor, it is easy to show that the MSE is a nondecreasing function of p [24]; under ideal conditions, for an autoregressive signal of order q,

the curve obtained by plotting the MSE as a function of p remains constant for $p \geq q$. In general, for any dataset, the curve flattens out as p grows and a flatness test has indeed been suggested in the past as the method to determine the correct order for the predictor [26]. It is important at this point to introduce the notion of *cost* of the LP representation; we can measure this cost by the number of bits needed to encode (and transmit) the predictor's parameters and, independently of the overall encoding strategy, this number is clearly a non decreasing function of the LP order. In modeling and compression application, the diminishing returns in terms of MSE with growing model order must be weighted against the corresponding increase in bit rate. An optimal tradeoff would deploy different orders to different portions of the signal so that each segment is coded with the same benefit.

If we now consider the composite LP analysis of a signal with respect to a given segmentation, the measure of accuracy of the modelization is given by the cumulative squared errors of the segmental predictors while the cost (in terms of encoded LP parameters) is a nondecreasing function of the sum of model orders and of the total number of segments. If we identify the squared error as the model's distortion and its descriptive cost as its rate, the optimal segmentation and the optimal order allocation can be determined jointly as the solution to a R/D optimization. Please note that true optimality can not be based on a local "stationarity" test nor on a local flatness test but only on a global minimization where the segmentation and the orders play the role of free variables.

4.1.2 R/D optimal Linear Prediction

Problem setup

With the notation for the segmentations and allocations defined as usual (section 2.2.4), assume there are K different LP models which could be applied to a segment; these are essentially predictors of different orders, but could include predictors whose parameters are then quantized and coded in different ways. For a given segmentation $\boldsymbol{t}$ and a related allocation $\boldsymbol{w}$ we can write as usual

$$D(\boldsymbol{t}, \boldsymbol{w}) = \sum_{i=1}^{\sigma(\boldsymbol{t})} d(t_i, t_{i+1}; w_i) \tag{4.6}$$

where in this case $d(t_i, t_{i+1}; w_i)$ is the squared error as in (4.4) for a linear predictor of order w_i applied to $x_{t_i}^{t_{i+1}-1}$. Similarly, we will assume that the LP coefficients are coded independently and that their cost in bits is a function $r(\cdot)$ of the predictor's order k, including the side information necessary to identify the order, the segment's length and possi-

bly the quantization scheme. We will then write

$$R(\boldsymbol{t}, \boldsymbol{w}) = \sum_{i=1}^{\sigma(\boldsymbol{t})} r(w_i). \tag{4.7}$$

Implementation

The LP "engine" will be Durbin's algorithm [6], an autocorrelation method. Although no tapering data window will be employed on the segments, the adaptive segmentation is sufficiently flexible to minimize boundary effects and the optimal solutions does not significantly differ from what would be obtained using the covariance method; the autocorrelation method simply puts a slightly higher bias on longer segment lengths. The following derivation is primarily illustrative of the key algorithmic issues, while still computationally efficient; subtler implementations can be envisaged, possibly involving more sophisticated covariance schemes [32].

While the segmentation algorithm allows for an extremely fine resolution, with minimum segment size of one data point, in most application it is more appropriate to allow for a coarser granularity, with segment sizes multiple of a C-point interval. Call this minimal span a *cell* and define $\boldsymbol{y} = y_1^L$ the collection of cells corresponding to $\boldsymbol{x}$ (assume $N = CL$, possibly by zero-padding). The optimal segmentation/allocation algorithm can therefore be applied to $\boldsymbol{y}$ with the substitution

$$d_{\boldsymbol{y}}(t_i, t_{i+1}; w_i) \leftarrow d_{\boldsymbol{x}}((t_i - 1)C + 1, (t_{i+1} - 1)C; w_i) \tag{4.8}$$

where the subscripts indicate the underlying signal. In the following, the subscript $\boldsymbol{y}$ will be implicit.

Assume we allow for K possible linear predictor orders for the segments, $p_1 < p_2 < \ldots < p_K$. Since the Durbin's recursion computes the prediction error for all lower order problems as well, we will solve an order-p_K LP problem for all segments. This involves the computation of the sum-products (4.5) which, for the autocorrelation method, depend only on the lag $l = |i - j|$. With respect to $\boldsymbol{y}$, for cell n we have

$$\rho_n(l) = \sum_{h=Cn}^{C(n+1)-1-l} x_h x_{h+l} \tag{4.9}$$

and, for any interval y_n^m, $\rho_{[n,m]}(l)$ satisfies the following chain relation:

$$\rho_{[n,m]}(l) = \rho_{[n,m-1]}(l) + \rho_m(l) + \delta_m(l) \tag{4.10}$$

with

$$\delta_m(l) = \sum_{h=Cm-l}^{Cm-1} x_h x_{h+l}. \tag{4.11}$$

Trellis algorithm

Algorithm 2.2 can be customized as follows:

Algorithm 4.1: Linear Prediction

Step 1: Initialization

1 Compute for all cells $y_n, 1 \leq n \leq L$ both $\rho_n(l)$ and $\delta_n(l)$ for $0 \leq l \leq p_K + 1$ ($\delta_1(l) = 0$).

2 Create the trellis as usual;

3 For $n = 1$ to N

For each $s_{n,m} \in S_n$:

Associate to the state a set of autocorrelation estimates $\varphi_{n,m}(l) = \varphi_{n-1,m-1}(l) + \rho_n(l) + \delta_n(l)$ (with $\varphi_{\cdot,0}(\cdot) = 0$);

Solve Durbin's recursion from order p_1 to p_K using $\varphi_{n,m}$ and associate to $s_{n,m}$ the relative squared errors $d(n-m, n+1; p_i)$, $1 \leq i \leq K$.

From here the population, backtracking and iteration steps can proceed as usual.

As for the initial values of the Lagrange parameter, the minimum allowable distortion is achieved for $\lambda_{\min} = 0$. A good starting value for $\lambda_{\max}$ can be inferred from the following argument. For a rate of zero bits, since no prediction is made, the resulting distortion is equal to the energy of the entire signal:

$$D_0 = \sum_n x_n^2. \tag{4.12}$$

As we now sweep λ from $+\infty$ to 0, consider the first value λ_1 for which $r^*(\lambda_1) \geq 1$ bit. Because of (2.17), for all (R, D) pairs it is $D + \lambda_1 R \geq d^*(\lambda_1) + \lambda_1 r^*(\lambda_1)$ and, in particular, for $(0, D_0)$ we can write

$$\lambda_1 \leq \frac{D_0 - d^*(\lambda_1)}{r^*(\lambda_1)} \leq D_0 \tag{4.13}$$

We can therefore set $\lambda_{\max} = D_0$.

Computational requirements

The computational requirements for the algorithm can be broken up as follows. The initialization step requires $O(N)$ adds and multiplies for the computation of the autocorrelation estimates. Then, for each state in the trellis, we need to sum up the cumulative estimates ($O(p_K)$ adds and multiplies) and solve a LP problem with Durbin's algorithm ($O(p_K^2)$ adds and multiplies plus $O(p_K)$ divisions). The number of states is $L(L+1)/2$, $L = N/C$, so in total the initialization step requires $O((N/C)^2)$ operations. Each iteration over λ requires a maximum of p_K comparisons (plus two adds and one multiply) per state, with again a total of $O((N/C)^2)$ operations. Storage is proportional to the number of states.

For large datasets, as in the case of speech, the algorithm can easily be applied to data segments separated by clearly identifiable features such as silence or gaps, for which an a-priori allocation decision can be made with little risk of suboptimality. Alternatively, since backtracking from time index $n < L$ yields the optimal segmentation for y_1^n, a splitting point can also be determined heuristically by executing the backtracking step for all n's: once a point n_0 of stable path convergence is found, in the sense that at least a minimum number of successive backtracked paths converge in n_0, the trellis can be restarted at that point and the allocation for $n < n_0$ determined independently of successive data. Again, the incurred suboptimality is minimal, but these solutions requires an explicit splitting of the total rate budget which might be difficult to determine a priori.

4.1.3 Results

We can finally compare the results between a fixed-window coding strategy such as that employed in the LPC-10 speech coder [58] and the R/D optimal LP coding. The LP block in the LPC-10 coder splits the data into fixed 22.5 ms frames (180 data points for 8 KHz sampled speech) and computes an order-10 predictor for each frame. In our coder, we set $K = 6$, allowing the order to span the range from $p_1 = 5$ to $p_6 = 10$, and we set the cell size to $C = 60$ samples. Quantization follows the LPC-10 specifications, with 5 bits for the first four coefficients, 4 bits for the next four, 3 and 2 bits for the last two; the LP parameters are in the form of reflection coefficients and therefore the effects of quantization can be easily integrated in Durbin's recursion. The cost of side information, which specifies the length of the current segment, is set to 10 bits per segment, with three bits for the predictor's order and seven bits for the segment's length. The algorithms are applied to a 2.2 seconds speech signal from a standard test corpus with DC bias removed and normalized to unit amplitude.

Figure 4.2 shows the set of solutions obtained by iterating the algorithm between $\lambda_{\min}$ and $\lambda_{\max}$; the distortion is the cumulative squared error normalized by the signal's energy, while the rate is normalized by the length of the speech sample. The circle

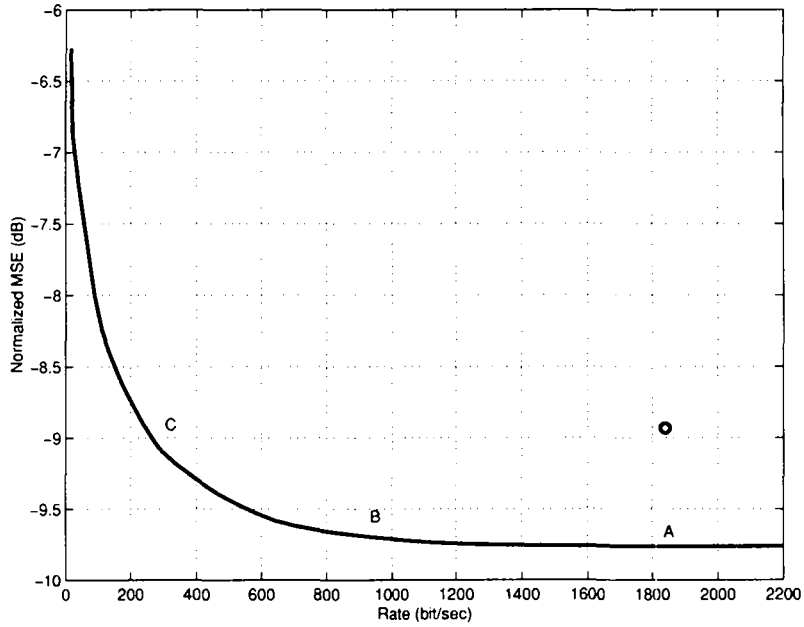


Figure 4.2: Operational Rate/Distortion curve for a speech segment; the dot indicates the operational R/D point of the fixed-window, LPC-10 predictor.

shows the operating (R, D) point for the fixed-window algorithm at its constant bitrate of 1822 bit/sec; the normalized error power is -9.02 dB. At an equivalent rate, the R/D optimal algorithm gains 0.74 dB; more significantly, an equivalent error power of -9.4 dB is achieved at a bitrate of 277 bit/sec, which is approximately six times lower (point C in Figure 4.2). Figure 4.3 show the segmentations and allocations relative to points A, B and C in the R/D curve alongside with the speech signal. The width of a block represent a segment's length, and its height the corresponding LP order. At first it might seem surprising that a coarse segmentation such as C has the same MSE characteristics as the fine windowing of the LPC-10 scheme; yet, while in both cases most of the signal's energy is still in the residual, its time distribution is very different. The LPC-10 22.5 ms interval is appropriate as a baseline grid to resolve fast transients but offers no gain with respect to longer range prediction; under the same distortion constraint, the dynamic segmentation algorithm still resolves the main transients, but avoids isolating small speech portions, which cannot be modeled by an order-10 predictor anyway, in favor of a more accurate estimation of the quasi-stationary parts.

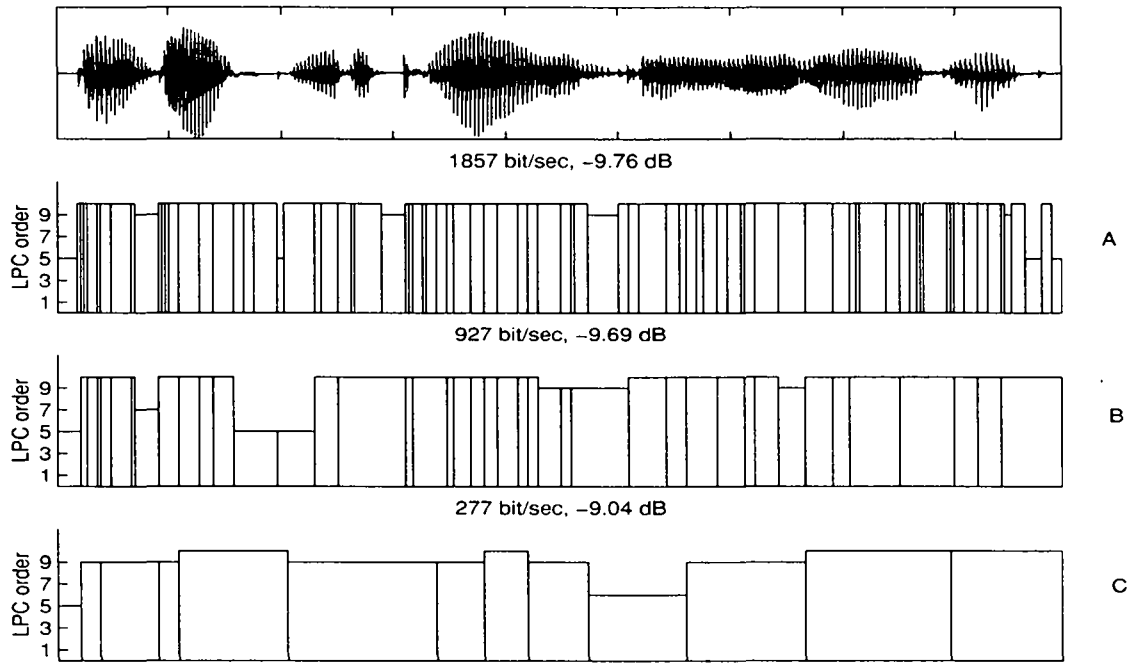


Figure 4.3: Segmentations and allocations for the speech signal in the upper panel at the (R, D) operational points marked A, B, and C in Figure 4.2.

4.2 Arithmetic coding

Several picture coding systems like JBIG, JPEG, and EZW, utilize or allow for a binary arithmetic coder as their last processing step. Indeed, as we mentioned in the beginning, given an accurate estimate of the probability distribution of the binary input data, arithmetic coding produces a code stream whose expected length is within two bits of the source entropy. The practical difficulty rests with the estimation of the source statistics, and much more so in the case of nonstationary data; in this case, adaptive schemes are a need.

Again, adaptivity in the probability estimation process can be of two kinds. *Backward adaptivity* infers from past data a statistical description which is extended to the current input. The Q-Coder [40], for instance, implements a backward predictive model by means of a state machine whose transition table is carefully tailored to the average statistical behavior of bilevel images. The very structure of the adaptation mechanism, however, imposes a constraint on the generality of the predicted distribution; furthermore,

causal transition in the data (as in a piecewise stationary memoryless source for which the underlying probability distribution changes abruptly in time) will cause a substantial mismatch overhead in any backward adaptive coder.

Forward adaptivity, on the other hand, relies on both past and future data to generate an estimate of the underlying statistical model; two-pass, adaptive Huffman coding in JPEG is an example [60]. Clearly, this model offers complete generality in that an exhaustive minimization process can be performed to single out the distribution which minimizes the length of the coder's output. Apart from the higher computational cost of such a procedure, however, the price to be paid for this generality with respect to backward adaptive systems consists in the *side information* which needs to be tagged to the coded data to inform the decoder of the selected statistical model.

The choice of the most advantageous estimator depends entirely on the particular data sequence which has to be encoded: it is easy to produce data sets for which a backward adaptive model outperforms a forward adaptive model, and vice-versa. Therefore, the criterion for choosing the type of estimator can only be the compression ratio at the encoder's output. For piecewise stationary inputs, this approach leads to a joint search for the segmentation and the corresponding sequence of backward and forward models which minimizes the total rate. We are back to our usual R/D optimization scenario, with just a little difference: since the compression is lossless, the distortion is always zero, and it will not enter in the minimization process; we could look at this also as a case in which $\lambda = +\infty$. It must be remarked, however, that the rate is nonseparable, due to the adaptive nature of the coding models.

4.2.1 Theoretical background

For a wide class of *stationary* signals including Markov processes and, as a particular case thereof, Bernoulli sequences, Minimum Description Length (MDL) theory [48] provides a lower bound on the performance of any compression scheme. Specifically, given a *finite* N-point data sequence x_1^N, the performance of any coder, backward or forward adaptive, is bounded by the *information* of the sequence, defined as

$$I(\boldsymbol{x}) = \min_{d} \min_{\boldsymbol{\theta}} \{ -\log P_{\boldsymbol{\theta}}(\boldsymbol{x}) + (1/2) d \log N \}, \tag{4.14}$$

where the (generic) statistical model $P_{\boldsymbol{\theta}}(x)$ depends on the d-element parameter vector $\boldsymbol{\theta} = \theta_1^d$ and the minimization is carried out over all number of parameters and all corresponding parameter vectors.

Consider the case of Bernoulli sources for example, for which the parameter vector is just the probability θ of, say, a '1'; by taking expectations over all N-point data se-

quences we obtain the average information

$$I(\boldsymbol{x}) = H(\theta) + \frac{1}{2}\log N, \tag{4.15}$$

where $H(\cdot)$ is the binary entropy function. The last term represents the cost of not knowing θ a priori, and it can be paid for in two ways: in forward adaptive coders, by providing the estimated value for θ to the decoder; in backward adaptive coders, by the estimation mismatches in the causal prediction.

For nonstationary sequences the situation is more complex; the previous bounds in a MDL sense have been extended to *piecewise stationary* Bernoulli sources by Merhav [29]. Intuitively, in this case the extra cost to be paid also includes the information pertaining to the parameter transitions in the data sequence; it can indeed be shown that for N-point sequences containing M switchpoints, the average information can be expressed as:

$$I(\boldsymbol{x}) = \frac{1}{M}\sum_{i=0}^{M} l_i H(\theta_i) + M\frac{1}{2}\log N + M log N, \tag{4.16}$$

where l_i is the length of the data segment for which the Bernoulli parameter stays constant and equal to θ_i, and $\sum l_i = N$. With respect to (4.15), the cost associated to the Bernoulli parameter is multiplied by M; furthermore, an additional price of $\log N$ bits for each transition appears. Merhav also proved that this lower bound can be attained by a purely sequential coder (no look-ahead), but the method is hardly practical. These results were recently extended by Willems [63], who addressed more practical issues in the problem of determining the set of parameters for a piecewise-constant Bernoulli sequence. His analysis is however mostly concerned with the attainability of the Merhav bound in a strictly sequential fashion.

While these lower bounds are very useful in stating the best performance one can expect from a coding system, their asymptotic nature makes them less helpful in the case of a single realization of a nonstationary process. In particular, the theoretically equivalent efficiency of most estimators holds only in the limit; for one finite data sequence, practical and computationally efficient estimators can yield substantially different results. Here, we somewhat step aside from the theoretical framework described above; rather, our goal is to implement a practical system using widely available building blocks (such as the Q-coder), the baseline performance of which is very well known in a variety of settings. The aim is twofold; on one hand we want to provide a system which easily "tags on" to pre-existing coding scheme; one might indeed want to retain the core structure of such coders for efficiency or compatibility reasons. Secondly, we want this system to be as open as possible; the immediate advantage of this is that it is straightforward to augment the family of coding models to arbitrarily complex predictors. This is especially useful

when the data are no longer iid over some segments, and context-based prediction (which takes into account higher-order statistics) becomes necessary.

4.2.2 Rate-optimal arithmetic coding

Problem setup

We will apply the optimal segmentation/allocation algorithm to the arithmetic coding problem with respect to two coding models ($K = 2$). The first model is the backward adaptive predictor implemented in the Q-Coder [40, 30]. This estimator declares one of the two possible input values (say 0) the *least probable symbol* (LPS) and assigns a probability to it; the other input value (1) is labeled the *most probable symbol* (MPS). The LPS probability is constantly adjusted depending on the sequence of LPS's and MPS's which are encoded; if the LPS probability passes the 0.5 mark, the LPS and MPS labels are swapped. The probability estimation is carried out by means of a 29-value state machine; an LPS probability value and pointers to the next and previous states are associated to each state. The entries in the state machine were found experimentally in the context of bilevel image coding. At each point in time, the Q-coder internal status can be represented as a four-element vector γ consisting of the value for the LPS, the LPS probability, the value for the coding interval as stored in the internal register, and the value for the carry. This allows us to compute the output rate for all segment lenghts in an incremental fashion.

The second model is a forward adaptive predictor in which an estimate of the Bernoulli parameter for a segment x_m^n is simply

$$\theta = \frac{1}{n-m+1} h(x_m^n), \tag{4.17}$$

where $h(\cdot)$ indicates the Hamming weight of the sequence; this estimate (or its complement $1-\theta$ if $\theta > 1/2$) is subsequently quantized to the closest probability $\hat{\theta}$ in the Q-Coder table, so that it can be indexed by five bits. The data are then encoded with the Q-coder algorithm in which the adptive estimation is turned off; initially the LPS is set to 0 if $\theta > 0.5$ and to 1 otherwise, and the LPS probability is set to $\hat{\theta}$.

With the notation for the segmentations and allocations defined as usual (section 2.2.4), the optimization takes the form of (2.17), but since the distortion is zero, there is no need for a Lagrange parameter. However, this is a typical situation in which the cost of side information is crucial, and we will therefore iterate the solution over all possible costs for encoding a transition using the colored bits strategy of Section 2.3.2. In the worst case, the input stream is a maximum entropy sequence, and therefore the largest cost for a transition must be set to $\log_2 N$. We will denote the cost of a transition by c,

and the corresponding rate expansion factor by ρ; for a given choice of c the optimization problem becomes:

$$\min_{\boldsymbol{t}\in T} \min_{\boldsymbol{w}\in W(\boldsymbol{t})} \{R(\boldsymbol{t},\boldsymbol{w})\} \tag{4.18}$$

with

$$R(\boldsymbol{t},\boldsymbol{w}) = \sum_{i=1}^{\sigma(\boldsymbol{t})} (1+\rho) r(t_i, t_{i+1}; w_i) + c'(w_i) \tag{4.19}$$

where $r(n,m;k)$ is the size of the output stream when coding x_n^{m-1} with the backward adaptive ($k=1$) or the forward adaptive ($k=2$) coder, and the side information is

$$c'(k) = \begin{cases} c+1 & \text{if } k=1 \\ c+1+5 & \text{if } k=2 \end{cases}; \tag{4.20}$$

one bit is indeed necessary to specify the model and, in the case of the forward adaptive model, five more bits encode the Bernoulli parameter.

Implementation

Just as in the speech coding example, the granularity of the segmentation can be modified by rearranging the input bitstream into contiguous multi-bit cells of C bits each; the new input vector will therefore be $\boldsymbol{y} = y_1^L$ with

$$y_n^m = x_{(n-1)C+1}^{mC} \tag{4.21}$$

and $N = CL$, possibly by zero-padding. The optimal segmentation/allocation algorithm can therefore be applied to $\boldsymbol{y}$ with the tacit assumptions that all parameters and rates for any segment of $\boldsymbol{y}$ are in fact computed for the underlying signal $\boldsymbol{x}$ using the above relation.

Trellis algorithm

Algorithm 2.2 can be adapted as follows:

Algorithm 4.2: Arithmetic coding

Step 1: Initialization

Each state will hold the following quantities: $h_{n,m}$, the cumulative Hamming weight of the segments, $\gamma_{n,m}$, the latest internal configuration of the Q-coder, and the rates for the backward and forward coding schemes. The initialization proceeds incrementally:

1 For $n = 1$ to N

For each $s_{n,m} \in S_n$, $1 \leq m \leq n$

Associate to the state the cumulative Hamming weight of the segment: $h_{n,m} = h_{n-1,m-1} + h(y_n)$ (where $h_{n,0} = 0 \forall n$).

Compute $r(n - m + 1, n; 1)$ for the backward model: set the initial Q-coder configuration to $\gamma_{n-1,m-1}$ and run the coder over y_n. Associate $\gamma_{n,m}$, the resulting Q-coder configuration, to the state.

Compute $r(n - m + 1, n; 2)$ for the forward model: determine the Bernoulli parameter as $\theta_{n,m} = h_{n,m}/Cm$; quantize $\theta_{n,m}$ over the Q-coder probability table and estimate the rate as $(Cm)H(\hat{\theta})$.

2 Finally, create an extra state set $S_{L+1} = \{s_{L+1,1}\}$.

Step 2: Trellis path population

3 Select a value of c and determine the corresponding $\rho = -\log_2(1 - 2^{-c})$; let $j_0^* = 0$.

4 For $1 \leq n \leq L$

Determine the minimum cumulative rate:

$$j_n^* = \min_{1 \leq m \leq n} \min_{k=1,2} \{j_{n-m}^* + (1 + \rho) r(n - m + 1, n; k) + c'(k)\}$$

using the values stored in the trellis. Assume the minimum is for the pair (m^*, k^*): connect s_{n,m^*} to $s_{n+1,1}$ and associate (j_n^*, m^*, k^*) to the path. Also, connect $s_{n,m}$ to $s_{n-1,m-1}$ for $m > 1$.

Step 3: Backtracking

See Step 3 in Algorithm 2.2.

Step 4: Iteration over c

Unfortunately, the relation between c and the final rate is not monotonic, and the iteration over the side information cost should in principle be exhaustive, from $\lceil \log_2 N \rceil$ to 2 bits. In practice, fewer iteration around the value of $\lceil (1/2) \log_2 N \rceil$ usually suffice.

Step 5: Final encoding

Once the optimal segmentations have been determined, the segments can be encoded accordingly; the segmentation structure and the sequence of models can be encoded by means of control codes in the binary output stream [2]. Here we choose to blend the side information within the stream by means of a three-symbol arithmetic coder in which a colored bit has probability $\epsilon = 2^{-c}$ and input symbols 0 and 1 have both probability $(1 - \epsilon)/2$. Each time a new segment begins, a colored bit is coded, followed by a zero or a one to indicate the backward or the forward model respectively; in the forward adaptive case, five more bits are also encoded to specify $\hat{\theta}$.

Computational requirements

The computational requirements for the algorithm are as follows. The initialization step requires a constant number of operation per state, proportional to the length of the cell C. Again, the number of states is $L(L+1)/2$, $L = N/C$, so that the initialization step requires a total of $O((N/C)^2)$ operations. Each iteration over c requires a number of comparisons proportional to the number of states, with once again a global cost of $O((N/C)^2)$.

It is to be noted that the complexity of the decoder is equivalent to that of a standard arithmetic decoder, and that the decoding is instantaneous. This complexity asymmetry, together with the encoder's inherent delay, suggest that the proposed algorithm is most useful in lossless data storage settings in which the premium is almost exclusively on compression ratios.

4.2.3 A simple example

In the following example, the input to the dynamic arithmetic coder is the raster scan of the bilevel image displayed on the left of Figure 4.5. While this is not, of course, a proper image coding system (no image-specific prediction is applied) it illustrates in an intuitive way the properties of the proposed algorithm. Figure 4.4 shows the achievable compression ratios as we vary the cost of a colored bit from 5 to 16 bit; the dashed line represents the baseline compression ratio obtained with the standard Q-coder. The shape of the curve is typical to most compression problems: when the cost of a colored bit is too low, the associated expansion factor dominates and the performance worsens significantly; the curve also shows the non-monotonic dependence of the compression ratio on the cost

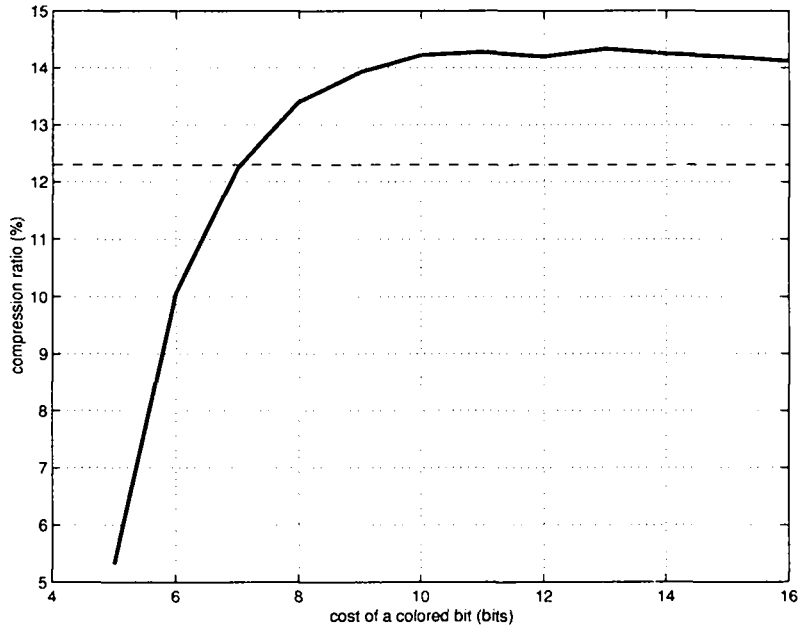


Figure 4.4: Compression ratio vs. cost of a colored bit

of side information, which requires an exhaustive search over the parameter space. In this case in particular, the optimum is achieved for a colored bit cost of 13 bits. The plot on the right of Figure 4.5 displays the LPS probability for the binary arithmetic coder versus the index of the coded byte for this latter case. The rectangular blocks in solid line represent portions of the signal for which the coder has identified a constant parameter for the LPS probability by a forward adaptive estimate; the dashed line represents the evolution of the LPS probability when the system works in a backward adaptive mode. We can identify four distinct segments, corresponding to the four textures in the image. Section A and D simply code the gray textures according to their pixel densities; forward adaptation is efficient here due to the regularity of the pattern. In section C, for which the image has the same number of black and white pixels per square region, the system locks to a LPS probability of 1/2, to avoid the coding inefficiency of an oscillating backward adaptation. Finally, section B is coded mainly in backward adaptive mode, which is consistent with the frequent and short transitions in the image. It is to be noted how the algorithm achieves a very good segmentation of the image. The compression rates are 14.3% for the proposed algorithm versus 12.3% for the Q-Coder.

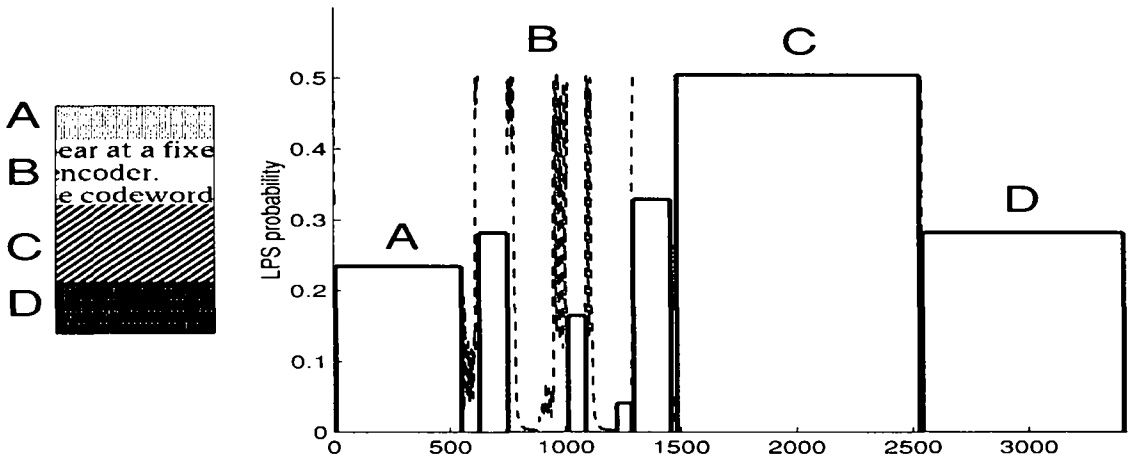


Figure 4.5: Arithmetic coding example

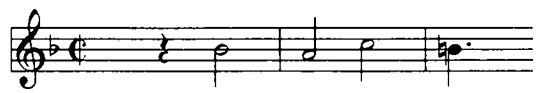
4.3 Summary

This chapter introduced two practical applications of the optimal segmentation/allocation framework. Section 1 described a linear prediction algorithm in which the usual fixed-size time windowing is replaced by a dynamic segmentation algorithm which allocates different order LP predictors to different portions of the data according to their local stationarity characteristics; the fundamentals of linear prediction, and of speech linear prediction in particular, have been reviewed in subsection 1.1, while subsection 1.2 extended the LP algorithm to include dynamic segmentation using the dependent allocation template; finally, in subsection 1.3 a practical implementation of the algorithm as an extension of the standard LPC-10 speech coding system has been described, with a discussion on the experimental results.

Section 2 applied the dynamic segmentation concept to binary arithmetic coding. The principles of adaptive arithmetic coding were reviewed in first two subsections, with a brief detour into the theoretical facets of coding limits for stationary sources. Subsection 2.2 placed the arithmetic coding problem in the context of rate optimality and introduced a dynamic algorithm in which the coder can switch between a backward (Q-Coder) and forward (counting) probability estimation mechanisms. Finally, some experimental results were described.

Chapter 5

Communications: Data Hiding for Audio Signals



– J. S. BACH, *Kunst der Fuge, BWV 1080.19*

Another fruitful domain of applicability for the optimal segmentation techniques introduced so far is data communication. Nonflat, time-varying communication channels are usually dealt with by invoking ergodicity and by designing a signaling system tuned to the time-averaged characteristics of the medium. In most cases, this is really the best that can be done, since a physical channel is something we have little or no control over and since real-time communication is the one goal. Yet there are a few particular instances where we can claim to possess perfect knowledge of past, present, and future channel conditions. One such instance is data hiding, where the "channel noise" is represented by a host signal to which arbitrary data are to be added in an undetectable way. A data hiding system distributes the aggregate of the hidden data in time and in frequency in order to exploit all the "energy gaps" in the host signal; if the latter is known in advance, this time-frequency distribution can be carried out by means of optimal allocation techniques, as we will show in the following.

5.1 Data hiding

Data hiding, or steganography, is concerned with embedding data into a "host" message (the *cover message*) in a way which is undetectable to an external observer; in this sense, it differs from cryptography (although the hidden data can itself be encrypted) since the cover message remains unaltered and entirely meaningful from a perceptual point of view. In recent years, there has been a lot of interest in steganographic techniques in connection with watermarking for copyright protection of audio and video material [1]; in these cases the goal is to embed a marker or a digital signature in proprietary material in order to track potential copyright infringements. Clearly, this requires that the data hiding method be extremely resilient to common data processing techniques such as compression, filtering, or cropping, which are viewed by the copyright owner as "attacks" on the watermark. Intuitively, it is easy to see that the capacity of the steganographic channel and its robustness to attacks are inversely related to each other; in watermarking applications this should not be a fundamental limitation since the amount of data to be hidden is relatively modest, yet it is still an active area of research to find a universally robust watermarking technique [11].

There is however another situation which can make good use of steganographic techniques, and that is the case when different data sources share the same, fixed physical channel. Multiple access channels are extremely well studied in the case of mobile communications, for instance, and we will see that some techniques that are commonly employed in dealing with fading or nonflat multiple access links can be of help in designing a data hiding system. However, the fundamental difference between such communication protocols and a data hiding scenario is that, in the latter, one particular data source takes priority with respect to the others; furthermore, some of the receivers might not have the ability to extract the hidden message, yet the composite data stream should be entirely equivalent (in some perceptual way) to the prioritary data stream alone. Everyday examples of such a scenario are for instance a color TV signal, which can be reproduced by older black and white sets; or teletext messages, inserted at the blanking intervals between frames in a way that does not affect the standard TV decoding process. Both these methods work by exploiting gaps in a decoding protocol; more sophisticated techniques can exploit the *perceptual* gaps in the human visual or auditory system instead, much in the same fashion as standard compression algorithms do. The common strategy, in all cases, is to maximize the amount of information that can be hidden in a given signal while preserving its perceptual properties. This maximization of throughput necessarily implies a very low resilience to signal manipulation, which sets us apart from watermarking techniques; at any rate, the goal is not secrecy or robustness towards attacks but the expansion (or better, the splitting) of the communication channel throughput in a "backward dependent" way.

In this chapter we will discuss a data hiding technique for digital PCM audio, based on the masking properties of the human auditory system (HAS). The setting might be that

of a compact disc (CD), for instance, to which we want to tag additional data such as lyrics or pictures without altering either the format of the support (the "Red Book" protocol) or its audio capacity (74 minutes approximately); the resulting CD will therefore be entirely compatible with standard CD players, will play to the maximum allowable duration, and yet will allow more sophisticated equipment to retrieve the hidden data. Some commercial examples which implement such an idea (albeit in minimal form) to improve audio quality are already available [37]; we will show that, by casting the data hiding technique in a rate/distortion framework, the optimal compromise between throughput and perceptual distortion can be achieved for all types of audio signal and for any data hiding protocol.

5.2 Perceptual audio coding

State of the art audio coding algorithms such as MPEG or Dolby's AC3 can provide acoustically transparent compression ratios of the order of 4:1 to 6:1 by cleverly exploiting the *masking phenomena* inherent in human hearing [31, 23]. Simply stated, masking occurs when weaker signal components are made inaudible by the presence of louder components; such weaker components are said to lie below the *masking curve* of the signal.

Compression algorithms quantize the signal so that the bulk of the overall quantization noise is hidden below the masking curve, and is therefore inaudible. In many audio steganography techniques, the hidden data source can be modeled by an equivalent additive noise generator; by shaping the power spectrum of such noise so that it lies below the masking curve, one could in theory achieve a perfectly transparent embedding. Perceptual hiding according to this general line has been indeed proposed in the context of audio watermarking [5]; in this paper we are however concerned with a tagging system for which maximum throughput and perfect extraction at the receiving end are the fundamental targets, which requires a finer exploitation of the masking properties of a signal and a more sophisticated hiding strategy.

5.2.1 Psychoacoustic modeling

In this section we will briefly review the fundamentals of psychoacoustic modeling from an operative, computational point of view; this simplified approach neglects some finer aspects of the hearing mechanism such as temporal masking but proves entirely adequate for the task at hand. From a physiological perspective, acoustic masking is a consequence of nonlinear processing in the inner ear; frequency-selective areas in the cochlea, called *critical bands*, exhibit a saturation characteristic whereby a loud sound component (the *masker*) renders inaudible all the weaker components in the same critical band which lie below a certain power threshold. The threshold is in fact a function of frequency and it

decays rather rapidly in an interval centered around the masker; its magnitude depends on the critical band number, on the power of the masker, and on the type of masker (whether an isolated spectral line or a noise-like component).

A psychoacoustic model tries to reproduce algorithmically these hearing mechanisms in order to obtain an estimate of the effectively inaudible portion of a given audio signal. Much study has been devoted to the characterization of masking functions [31], and different psychoacoustic models differ essentially in the parametrization of their shapes and decay rates. In the following we will rely on a simple model in which masking functions are approximated by piecewise linear functions in the log-power, bark frequency domain [38]; a unit of one bark corresponds to the width of a critical band and, since the width of successive critical bands increases by approximately a third of an octave, there is essentially a logarithmic mapping between bark scale and linear frequency scale. For complex tones the masking phenomenon is distributed across the entire audible spectrum, since each spectral component originates a local masking function; the sum of all masking thresholds for all components across the signal's bandwidth yields the overall *masking curve.*

An algorithmic process estimating the masking curve for a given signal can be illustrated with reference to the MPEG standard Psychoacoustic Model 1, which will also be used in section 5.4.2. It comprises the following steps [21]:

- *Computation of the power spectrum;* this is performed by a short time Fourier transform analysis.
- *Separation of tonal and non-tonal components;* since the masking power of isolated spectral lines is less than that of noise-like spectral components, the former are separated from the latter.
- *Computation of the individual masking thresholds;* this step is accomplished by convolving each spectral component by the appropriate (tonal or non-tonal) masking function.
- *Computation of the global masking curve;* the masking curve is obtained as the sum of the individual masking thresholds.

An additional step is required to map the masking curve thus obtained back to the linear frequency domain; the final result will be denoted by the time-frequency function $\Theta(t, f)$. For a fixed time instant t_0, this function is one realization of the masking curve; for a fixed frequency f_0, it represents the evolution of the masking threshold over time for the given frequency. Figures 5.1 and 5.2 display typical instances of a masking curve and of a time-varying masking threshold (at low frequency) respectively; here and in the following, the audio data are obtained from a string quartet CD [50].

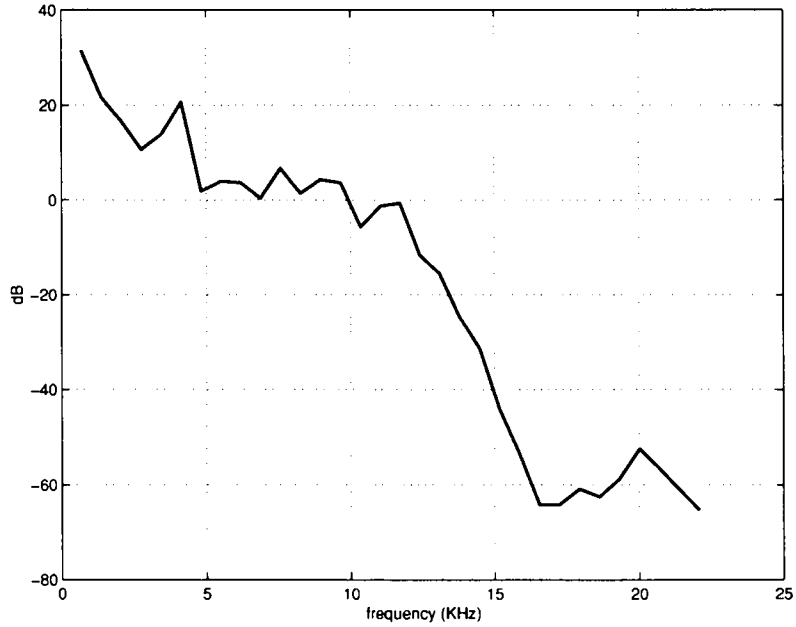


Figure 5.1: Masking curve for one frame (26 ms).

The time-frequency resolution of the psychoacoustic model depends on the underlying short time spectral analysis used to compute the power spectrum, and it is a trade-off between accuracy of the tonal and non-tonal representation and responsiveness to fast signal transients. In MPEG layer II, for an input sampling rate of 44.1 KHz, the psychoacoustic model produces a masking curve every 26 ms; in the following, we will call such an analysis interval as a *frame*.

5.2.2 Compression

As stated previously, the fundamental coding step of a perceptual audio coder is to quantize the signal separately over different subbands so that the quantization noise level for each subband is less than the minimum value of the masking curve over the subband. This minimum value is generally referred to as the masking threshold for the subband. Without going into further details, the most important point is to remark that, due to this signal dependent quantization, part of the total bit budget for the coder must be spent to inform the decoder of the different subband levels.

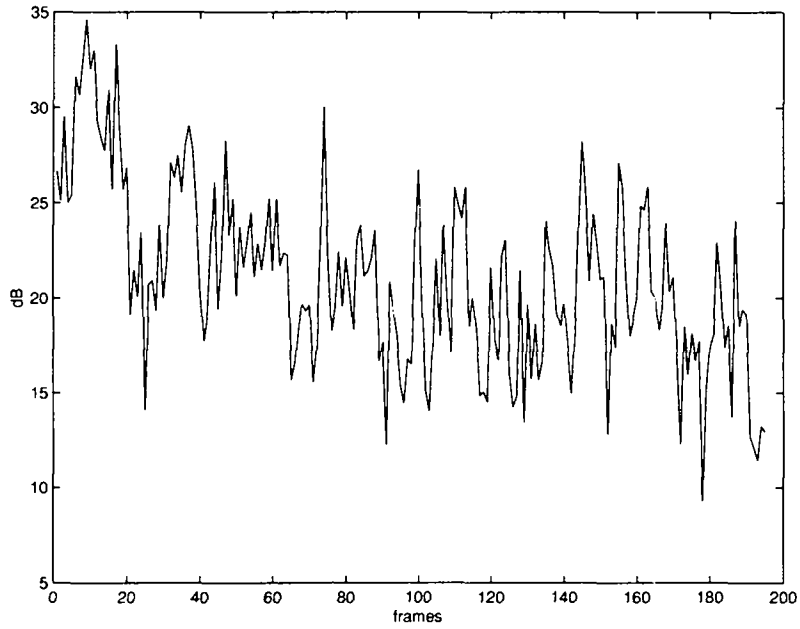


Figure 5.2: Time varying masking threshold for a frequency around 1 KHz.

5.3 Perceptual data hiding

Consider an audio signal $s(t)$, $0 \leq t \leq t_0$ bandlimited to f_N Hz: we have seen that the masking curve computation yields a function $\Theta(t, f)$, of which two typical time and frequency "slices" are displayed in Figures 5.1 and 5.2. This function represents a two dimensional power constraint for the transmission power which, in principle, should lie below the threshold [10]. In order to fulfill this constraint, we choose to adopt a separable approach by discretizing both the time and frequency axis; while this is admittedly not the most sophisticated way to achieve a time-frequency power shaping, it is in line with the current techniques which implement the psychoacoustic analysis model and allows for an easy integration of our transmission scheme in compression engines such as MPEG. In particular, spectral power shaping in frequency for a given time interval is accomplished by multicarrier modulation, which implements a discretized approximation of the reverse waterfilling algorithm [34]. The key point is that, if the entire frequency interval is split into M bands by a M-ary filterbank with perfect or almost perfect reconstruction capabilities, then *power shaping along the time axis can proceed independently for each filterbank subchannel.*

5.3.1 Discretization

Assume the (positive) frequency axis $[0, f_N]$ is subdivided into M adjacent, non overlapping bands $[f_m, f_{m+1})$, $f_0 = 0, f_M = f_N$: at any time instant t we can obtain M corresponding masking thresholds values by selecting the minimum value of $\Theta(t, f)$ in each subband. Also, the psychoacoustic model computes a full masking curve via short-time windowing over frames C samples apart; for a sampling frequency of f_s Hz, there are a total of $N = f_s t_0 / C$ frames, with $\Theta(t, f)$ constant for $(n-1)C/f_s \leq t < nC/f_s$, $n = 1, \ldots, N$. The completely discretized set of masking thresholds can therefore be denoted by the (discrete) function $\theta_m(n)$ for $m = 1, \ldots, M$ and $n = 1, \ldots, N$:

$$\theta_m(n) = \min_{f_{m-1} \leq f < f_m} \{\Theta((n-1)C/f_s, f)\}. \tag{5.1}$$

Using the same discretization grid, we would like to obtain a sequence of signal energy levels relative to the time-frequency tiles over which the masking thresholds are computed; these can be determined as:

$$\sigma_m(n) = \int_{(n-1)C/f_s}^{nC/f_s} |s(t) * g_m(t)|^2 dt \tag{5.2}$$

where the Fourier transform of $g_m(t)$ is the indicator function for the interval $[f_{m-1}, f_m]$; in practice, these energy levels can be computed as the average energies over a time interval of one frame at the outputs of a M-band filterbank.

As for the data tagging scheme, we assume we have M independent "transmitters" which operate over the $[f_{m-1}, f_m)$ bands; we also assume the power of the transmitters can be arbitrarily varied form one C-point interval to the next from a minimum level p_1 to a maximum level p_K. In the rest of the analysis, the actual transmission method is irrelevant as long as it can be modeled by a modulated, bandlimited additive noise source of equivalent power; we will also assume that the number of bits which can be successfully transmitted to the decoder each frame is a given function $r(p)$ of the transmission level p only.

5.3.2 Data hiding and side information

At first, one might think to use a sequence of transmission power levels which lies just below the sequence of masking thresholds for each subchannel; the receiver would redetermine the sequence of thresholds via a psychoacoustic model identical to the transmitter and decode the data. Unfortunately, this approach is not viable for two reasons. First, the masking curve computation for the audio signal and for the signal after the data has been embedded generally yields different results; this means that the decoder cannot recover the correct sequence of transmission levels, and therefore the data themselves. In addition, it might happen that the throughput requirements are stringent but cannot be

fulfilled by a sequence of power levels strictly below the thresholds; one might then decide to forgo the absolute undetectability constraint and use a sequence of levels which sometimes exceeds the masking thresholds. For both reasons, it is necessary to inform the receiver of the actual sequence of power levels, and this can only be done via side information. The situation is akin to perceptual compression where, in order to exploit the psychoacoustic gaps in the signal, side information about these gaps has to be supplied.

Since music can be reasonably modeled as a piecewise stationary signal, we design the transmission scheme so that side information is necessary only when the transmission power level changes in time. We therefore find ourselves in a dependent allocation situation, as illustrated in Section 2.2.3, and we can deploy the appropriate machinery to solve the problem. To recapitulate the conflicting requirements we are trying to fulfill, recall that side information necessarily uses up part of the overall throughput; the objective is thus to determine the transmission sequence which high throughput and a low description cost. At the same time, fewer power switches mean that we will overshoot the quickly varying masking threshold requirements more often. A tradeoff between throughput and perceptual distortion is now apparent: the goal is to determine the optimal transmission sequence with respect to these two parameters. The problem is again that of a rate/distortion optimization, where the maximization of the "rate" (the throughput) must be weighed against the corresponding increase in *perceptual* distortion.

5.4 R/D optimal data hiding

5.4.1 Problem setup

For a single subchannel, say subchannel m, the sequence of masking thresholds and signal levels for the time interval $[1, N]$ can be expressed in vector form as $\boldsymbol{\theta} = \theta_1^N$ and $\boldsymbol{\sigma} = \sigma_1^N$ (the channel subscript is inessential and is dropped throughout this section). An N-point allocation $\boldsymbol{w}$, defined the usual way, will represent a sequence of N transmission power levels between p_1 and p_K.

Call $D(\boldsymbol{w})$ the perceptual distortion we incur if we use the sequence of power levels $\boldsymbol{w}$ to transmit data over the subchannel; the corresponding net throughput will be denoted by $R(\boldsymbol{w})$. The goal is to maximize throughput while keeping the associated distortion below an acceptable minimum level D_0; this can be expressed as a constrained *maximization* problem:

$$\begin{cases} \max\limits_{\boldsymbol{w} \in W} \{R(\boldsymbol{w})\} \\ \\ D(\boldsymbol{w}) \leq D_0 \end{cases} \tag{5.3}$$

The above problem is similar, but not identical to (2.9); in order to reduce it to a more "standard" form we have to consider in more detail the structure of rate and distortion. The measure we choose for the distortion is the sum of the segmental perceptual noise to signal ratios (NSR), since the hidden data in each subband can be represented as an equivalent narrowband noise source.

A fundamental requirement is that the power of the noise be less than the power of the signal; then, if the noise level is below the masking threshold, the perceptual distortion is zero, otherwise its power is equivalent to the power of the transmitted data offset by the level of the threshold itself. We can therefore write

$$D(\boldsymbol{w}) = \sum_{n=1}^{N} d(n; w_n) \tag{5.4}$$

with

$$d(n;p) = \begin{cases} +\infty & \text{if } p \geq \sigma_n \\ 0 & \text{if } p \leq \theta_n \leq \sigma_n \\ \dfrac{p-\theta_n}{\sigma_n} & \text{if } \theta_n < p < \sigma_n \end{cases} \tag{5.5}$$

As for the throughput (rate), we can initially write

$$R(\boldsymbol{w}) = \sum_{n=0}^{N-1} r_c(n; w_n, w_{n-1}) \tag{5.6}$$

with $w_0 = p_1$ by convention and where $r_c(\cdot)$ is the net throughput associated to a given transmission level over one frame, minus the number of bits devoted to side information if a model switch takes place (see Equation (2.14)); we will assume that the cost of side information is always less than or equal to the minimum achievable frame rate for all frames and all models, so that all the terms in (5.6) are non negative. Assume now that the original (digital) audio channel allows for a maximum capacity of r_M bits per frame; for 16-bit PCM data, for instance, and a frame length of 36 samples , r_M would be 576 bits. Our goal is to split this capacity between an audio data channel proper and a tagged data channel so as to maximize the throughput of the latter. We can therefore introduce the following auxiliary quantity

$$\tilde{R}(\boldsymbol{p}) = N r_M - R(\boldsymbol{w}) \tag{5.7}$$

which represents the portion of the total capacity *not* devoted to the hidden data. It is easy to see that using $\tilde{R}$, the problem in (5.3) is equivalent to:

$$\begin{cases} \min\limits_{\boldsymbol{w}\in W}\{\tilde{R}(\boldsymbol{w})\} \\ \\ D(\boldsymbol{w}) \leq D_0 \end{cases} \tag{5.8}$$

and we are thus back to our familiar constrained minimization problem. From now on, the solution can proceed the usual way by casting the problem in unconstrained form using Lagrange multipliers and implementing Algorithm 2.1

Initial values for the iteration

Given the monotonic relationship between λ and $D^*(\lambda)$, an obvious choice for the initial minimum value is $\lambda_{\min} = 0$, for which the maximum allowable real throughput is achieved at the expense of a very large distortion. A good estimate for $\lambda_{\max}$ can be inferred from the following argument. Assume we have obtained (for some large value of λ) the solution pair $(0, \tilde{R}_{\max})$, at zero distortion. As we now sweep λ from $+\infty$ to 0, consider the first value λ_1 for which $D^*(\lambda_1) > 0$; a nonzero distortion means that the power of the data is larger than the masking threshold in at least one frame, and therefore

$$D^*(\lambda_1) \geq \min_{1\leq n\leq N}\{\min_{i|p_i>\theta_n}\{d(n;p_i)\}\} = D_{\min}; \tag{5.9}$$

$D_{\min}$ can be easily computed from a single pass over $\boldsymbol{\theta}$. Because of (2.10), for all $(D, \tilde{R})$ pairs it is $\tilde{R} + \lambda_1 D \geq \tilde{R}^*(\lambda_1) + \lambda_1 D^*(\lambda_1)$ and, in particular, for $(0, \tilde{R}_{\max})$ we can write

$$\lambda_1 \leq \frac{\tilde{R}_{\max} - \tilde{R}^*(\lambda_1)}{D^*(\lambda_1)} \leq \frac{Nr_M}{D_{\min}} \tag{5.10}$$

We can therefore set $\lambda_{\max} = Nr_M/D_{\min}$.

Multiple subchannels

The previous analysis easily carries over to the case of data hiding over all subchannels simultaneously. Optimality of the global rate/distortion tradeoff means that *all subchannels must operate at the same value for* λ since the overall rate and distortion are the (unweighed) sums of the rates and of the perceptual distortions for all channels. For the M different subchannels, by building

$$\boldsymbol{\theta} = \{\theta_1(1),\ldots,\theta_1(N),\theta_2(1),\ldots,\theta_2(N),\ldots,\theta_M(1),\ldots,\theta_M(N)\} \tag{5.11}$$

(and similarly for σ) the trellis algorithm can be used to obtained an NM-element allocation vector $\boldsymbol{w}$ from which the single subchannel allocations can be extracted in orderly fashion. A constant λ across subchannels implies an optimal distribution of the total amount of hidden data across channels, so that little or no data is tagged to frequency regions with poor masking capabilities such as in the high end of the audio spectrum.

5.4.2 Implementation

In the following section we will describe a practical implementation of the data hiding scheme for stereo, 44.1 KHz 16-bit PCM audio.

Signaling scheme

Discretization of the frequency axis is achieved by means of a 32-channel, uniform cosine-modulated filterbank as described in the MPEG Layer II specifications [21]; the width of a single subchannel is 690 Hz. The psychoacoustic model produces a masking curve every 1152 raw audio samples, which is then discretized to produce a masking threshold value for each subband; for a single subchannel, therefore, one threshold level is active over a frame of 36 subband samples.

Data tagging is achieved by replacing the least significant bits of the *subband* samples with data bits. Assuming the data has been scrambled, this is equivalent to an additive narrowband noise source with an approximate power of 6 dB per tagged bit; due to roundoff error in the analysis/synthesis filterbank computations, however, the least significant bit (LSB) of the subband samples is essentially random and therefore tagging can proceed safely only from the second LSB onwards. We allow for $K+1$ signaling models, where the model index identifies the number of tagged bits per frame from zero to K; the transmission power (in linear units) corresponding to k tagged bits per sample is then:

$$p_k = \begin{cases} 0 & \text{if } k = 0 \\ 4^{k+1} & \text{if } k > 0; \end{cases} \tag{5.12}$$

a signaling power of zero is needed when the finite distortion constraints cannot be met. Since there are 36 16-bit subband samples in a frame, $r_M = 576$ and $r(p_k) = 36k$; the minimum number of bits which can be tagged to a frame is therefore 36.

Side information is strictly related to the signaling protocol described below; in particular, the cost of a transition from zero bits to a nonzero level is higher than that for a transition to zero and the reason will be clear in the following section. In general, the cost for a transmission power switch from k to h bits/sample is

$$c(k,h) = \begin{cases} 0 \text{ bits} & \text{if } k = h \\ 36 \text{ bits} & \text{if } k \neq h \quad \text{and } k \neq 0 \\ 36h \text{ bits} & \text{if } k \neq h \quad \text{and } k = 0 \end{cases} \tag{5.13}$$

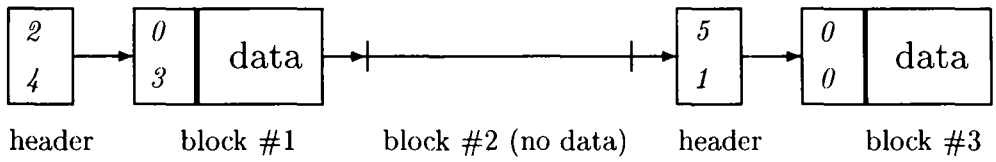


Figure 5.3: Signaling protocol for the allocation sequence $\boldsymbol{p}^* = \{l(2), l(2), l(2), l(2), l(0), l(0), l(0), l(5), l(5)\}$.

With these models, the optimal signaling sequence $\boldsymbol{w}^*$ is determined by the trellis algorithm, and tagging proceeds by passing the signal through the filterbank, replacing a number of LSB's in the subband samples with tagged data according to $\boldsymbol{w}^*$, and reconstructing the PCM audio signal with the complementary synthesis filterbank. At the decoder, after synchronization, an identical analysis filterbank recovers the subband samples and the data can be retrieved following the structure conveyed by the side information.

Signaling protocol

Synchronization between transmitter and receiver is established at the audio PCM sample level by inserting a sync sequence in the LSBs of the audio PCM samples; this has no noticeable perceptual effect on the audio material. The hidden data transmission protocol is displayed in Figure 5.3 for an allocation vector

$$\boldsymbol{w}^* = \{2, 2, 2, 2, 0, 0, 0, 5, 5\}.$$

First of all, the allocation vector is subdivided in blocks for which the number of bits per subband sample is constant; in this example there are three such blocks. Each tagged data block begins with the side information describing the number of bits/samples and the length (in frames) of the *next* block; for blocks which follow a zero bit allocation (and for the initial block) a separate one-frame side information header is necessary in which data is always tagged at one bit/sample by convention, since the decoding process must be restarted; in all other blocks side information is encoded at the current bit/sample allocation level.

Rather than optimizing the cost of side information, we chose to simplify matters by fixing the number of bits spent on signaling the lenght of a segment to 8. With this choice, the maximum block length is 6.6 seconds, which has proven largely sufficient in all practical cases; in the unlikely event of a longer block, the price to pay is an additional 36-bit side information slot after the 256th frame, which introduces a negligible suboptimality in

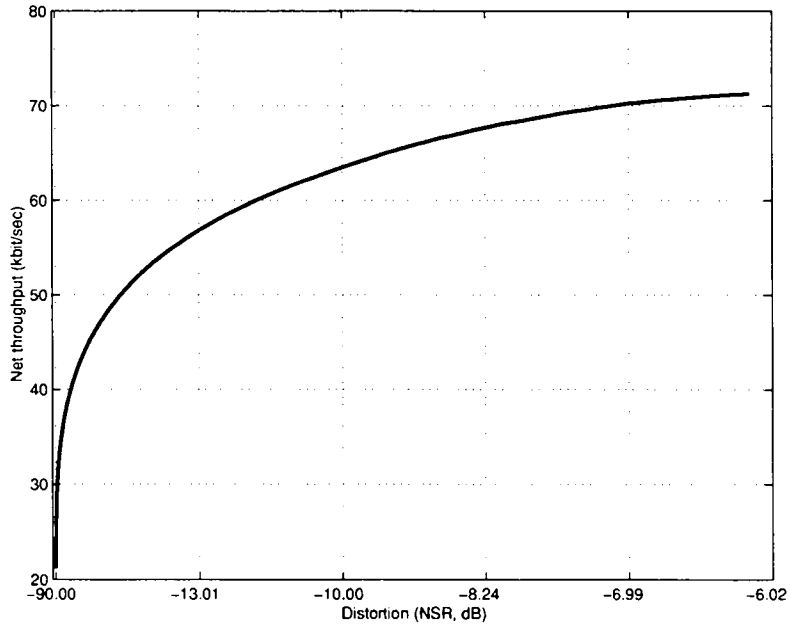


Figure 5.4: Throughput/distortion curve.

the global allocation. Four more bits completely specify the power level for a total allocation of 12 bits per transition; due to the catastrophic propagation of possible errors in the decoding of the allocation structure, a 3:1 repetition code is employed for a total of 36 bits.

5.4.3 Results

In Figure 5.4 the net throughput is plotted against the perceptual NSR for a 1 minute audio excerpt [50]. At zero distortion (noise always below the masking threshold) the net rate is about 21 kbit/sec; at twice this rate the average NSR grows to -25 dB, the limit after which, in most cases, distortion becomes disruptive. For the intermediate case of a 30 kbit/sec throughput and -35 dB NSR, Figure 5.5 displays how the R/D optimal algorithm allocates the transmission power across bands and in time for the first four subbands; the dotted line represents the signal power while the thin continuous line represents the masking threshold; the power of the data is plotted with a thick line.

In general, for stereo CD audio, capacities on the order of 30 kbit/sec can easily be achieved at no perceptual cost. The rate/distortion framework is completely general and can be adapted to many different signaling schemes; furthermore, since the structure of the tagged data is conveyed to the receiver by means of side information, the system is in-

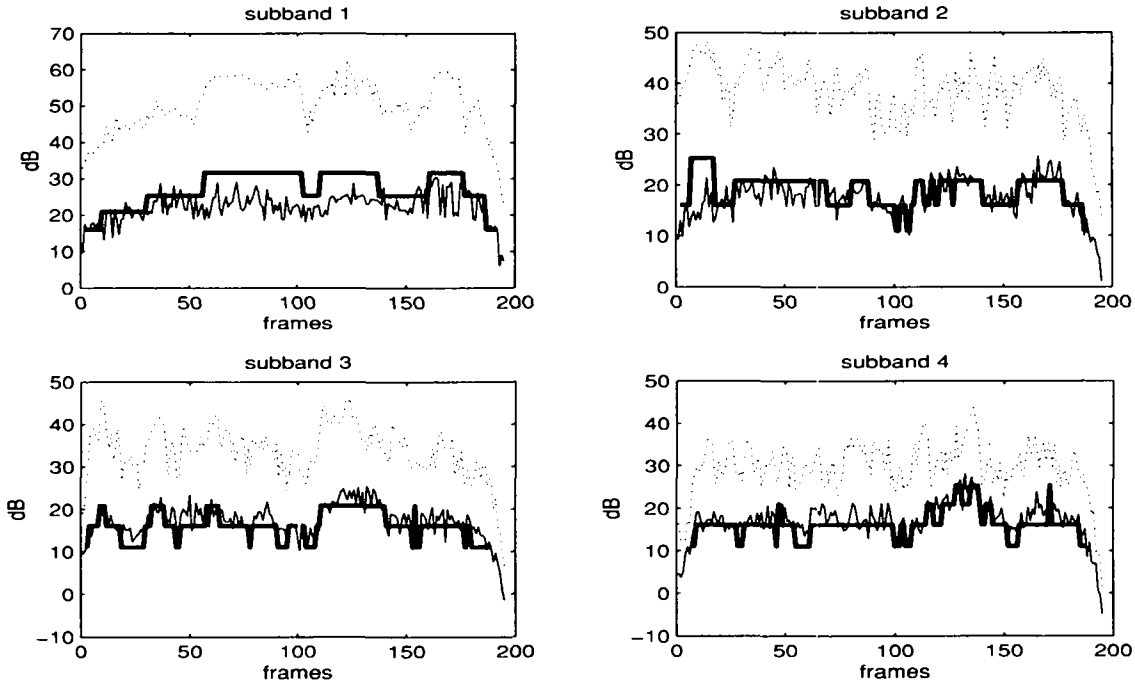


Figure 5.5: Transmission power allocations for the first four subbands.

dependent of the particular psychoacoustic model employed in the analysis.

5.5 Summary

This chapter introduced one further application of the optimal segmentation/allocation framework by developing an optimal data hiding scheme for CD-quality audio signals. The basics of data hiding have been illustrated in Section 1; Section 2 reviewed the fundamental notions of perceptual audio coding and psychoacoustic modeling, whose relevance to the problem at hand is discussed in Section 3. Section 4 introduced the R/D optimal algorithm, based on the partially independent trellis template; in this case the rate is the net rate of the hidden data while the distortion is identified with the perceptual noise introduced by the hiding process. Implementation details were also discussed in the same section, together with experimental results which attest a net data throughput for the data hiding scheme on the order of 30 kbit/sec at no perceptual loss.

Bibliography

[1] R. J. Anderson and F. A. P. Petitcolas. On the limits of steganography. *IEEE Journal on Selceted Areas in Communications*, 16(4):474–481, May 1998.

[2] R. B. Arps, J. M. Cheng, and G. C. Langdon. Control character insertion into arithmetically encoded strings. *IBM Tech. Disclosure Bullettin*, 25:2051, 1982.

[3] R. Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[4] T. Berger. *Rate Distortion Theory.* Prentice-Hall, Englewood Cliffs, NJ, 1971.

[5] L. Boney, A. H. Tewfik, and K.N. Hamdy. Digital watermarks for audio signals. In *Proc. of MULTIMEDIA '96*, pages 473–480, 1996.

[6] P. Clarkson. *Optimal and Adaptive Signal Processing.* CRC Press, 1993.

[7] A. Cohen, I. Daubechies, O. Guleryuz, and M. Orchard. On the importance of combining wavelet-based non-linear approximation in coding strategies. Manuscript, 1997.

[8] A. Cohen, I. Daubechies, and P. Vial. Wavelet bases on the interval and fast algorithms. *J. of Appl. and Comput. Harmonic Analysis*, 1:54–81, 1993.

[9] R. R. Coifman and M. V. Wickerhauser. Entropy-based algorithms for best basis selection. *IEEE Tran. on IT*, 38(2):713–718, March 1992.

[10] I. J. Cox, J. Kilian, T. Leighton, and T. Shamoon. Secure spread spectrum watermarking for multimedia. NEC Research Institue Technical Report 95-10, NEC Research Institure, Princeton, NJ, USA, October 1995.

[11] I. J. Cox and J.-P. M. G. Linnartz. Some general methods for tampering with watermarks. *IEEE Journal on Selceted Areas in Communications*, 16(4):587-591, May 1998.

[12] C. de Boor. *A practical guide to splines.* Springer, New York, NY, USA, 1978.

[13] B. Fette. High quality secure voice communication. *Speech Technology*, pages 40–48, October 1989.

[14] B. Fette and C. Jaskie. A 600 bps LPC voice coder. In *Proc. MILCOM '91*, pages 1215–1219, 1991.

[15] G.D. Forney. The Viterbi algorithm. *Proc. IEEE*, 61:268–278, March 1973.

[16] E. Bryan George, A. V. McCree, and V. R. Viswanathan. Variable frame rate parameter encoding via adaptive frame selection using dynamic programming. In *Proc. ICASSP*, volume 1, pages 271–274. IEEE, 1996.

[17] A. Gersho and R.M. Gray. *Vector Quantization and Signal Compression.* Kluwer, 1992.

[18] G. H. Golub and C. F. van Loan. *Matrix Computations.* Johns Hopkins, Baltimore, MA, USA, 1983.

[19] M. Goodwin. *Adaptive Signal Models.* Kluwer, Norwell, MA, USA, 1998.

[20] C. Herley, J. Kovačević, K. Ramchandran, and M. Vetterli. Tilings of the time-frequency plane: Construction of arbitrary orthogonal bases and fast tiling algorithms. *IEEE Tran. on SP*, 41(12):3341–3359, December 1993.

[21] ISO/IEC. *Internat. Standard IS 11172 (MPEG).* ISO, 1993.

[22] N. S. Jayant and P. Noll. *Digital Coding of Waveforms.* Prentice-Hall, Englewood Cliffs, NJ, 1984.

[23] J. D. Johnston. Transform coding of audio signals using perceptual noise criteria. *IEEE Journal Sel. Areas Commm.*, 6(2):314–323, Feb. 1988.

[24] A. M. Kondoz. *Digital Speech.* Wiley, Chichester, England, 1994.

[25] J. S. Lim and A.V. Oppenheim, editors. *Advanced Topics in Signal Processing.* Prentice Hall, Englewood Cliffs, NJ, 1988.

[26] J. Makhoul. Linear prediction: A tutorial review. *Proc. IEEE*, 63:561–580, April 1975.

[27] S. Mallat. *A Wavelet Tour of Signal Processing.* Academic Press, San Diego, CA, USA, 1997.

[28] S. Mallat and F. Falzon. Analysis of low bit rate image transform coding. *IEEE Tr. SP*, 46(4):1027–1042, Apr 1998.

[29] N. Merhav. On the minimum description length principle for sources with piecewise constant parameters. *IEEE Trans. on IT*, 39(6):1962–1967, November 1993.

[30] J. L. Mitchell and W. B. Pennebaker. Software implementations of the Q-Coder. *IBM J. Res. Develop.*, 32(6):753–774, November 1988.

[31] B. C. J. Moore. *Psychology of Hearing.* Academic Press, San Diego, CA, USA, 1997.

[32] M. Morf, B. Dickinson, T. Kailath, and A. Vieira. Efficient solution of covariance equations for linear prediction. *IEEE Trans. ASSP*, ASSP-25(5):429–433, October 1977.

[33] R. Bellman nad R. Roth. Curve fitting by segmented straight lines. *American Statistical Association Journal*, pages 1079–1084, September 1969.

[34] P.O. Okrah and J. M. Cioffi. Multichannel modulation as a technique for transmission in radio channels. In *Proc. Vehicular Technology Conference*, pages 29–33, Secaucus, NJ, USA, May 1993.

[35] A. Ortega and K. Ramchandran. Rate-Distortion methods for image and video compression. *IEEE Signal Processing Magazine*, Oct 1998.

[36] A. Ortega, K. Ramchandran, and M. Vetterli. Optimal trellis-based buffered compression and fast approximations. *IEEE Trans IP*, 3:26–40, 1994.

[37] Pacific Microsonics. Hdcd: High definition compatible digital. `http://www.hdcd.com`.

[38] D. Pan. A tutorial on MPEG audio compression. *IEEE Multimedia Journal*, Summer 1995.

[39] A. Papoulis. *Signal Analysis.* McGraw Hill, New York, NY, USA, 1977.

[40] W. B. Pennebaker, J. L. Mitchell, G.G. Langdon, and R.B. Arps. An overview of the basic principles of the Q-Coder adaptive binary arithmetic coder. *IBM J. Res. Develop.*, 32(6):717–726, November 1988.

[41] P. Prandoni, M. Goodwin, and M. Vetterli. Optimal time segmentation for signal modeling and compression. In *Proc. ICASSP*, volume 3, pages 2029–2032, Munich, April 1997.

[42] P. Prandoni and M. Vetterli. A mixed framework arithmetic coder. In *Proc. PCS97*, Berlin, 1997.

[43] P. Prandoni and M. Vetterli. Approximation and compression of piecewise smooth functions. *Phil. Trans. Royal Society London*, August 1999.

[44] P. Prandoni and M. Vetterli. R/D optimal linear prediction. Submitted to *IEEE Trans. SP*, 1999.

[45] K. Ramchandran and M. Vetterli. Best wavelet packet bases in a rate-distortion sense. *IEEE Tran. on IP*, 2(2):160–175, April 1993.

[46] K. Ramchandran and M. Vetterli. Rate-Distortion optimal fast thresholding with complete JPEG/MPEG decoder compatibility. *IEEE Tran. on IP*, 3(5):700–704, September 1994.

[47] E. Riskin. Optimal bit allocation via the G-BFOS algorithm. *IEEE Trans. IT*, 37(2):400–402, March 1991.

[48] J. Rissanen. Universal coding, information, prediction, and estimation. *IEEE Tran. on IT*, 30(4):629–636, July 1984.

[49] M. Schroeder and B. Atal. Code excited linear prediction: High quality speech at low bit rates. In *Proc. ICASSP*, pages 937–940, 1985.

[50] F. Schubert. String Quartet No. 13 in A Minor, Op. 29. *The Guarneri String Quartet*, 1997.

[51] X. Serra and J. Smith. Spectral modeling synthesis: A sound analysis / synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal*, 14(4):12–24, Winter 1990.

[52] C. E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. Journal*, 27, 1948.

[53] C. E. Shannon. Coding theorems for a discrete source with a fidelity criterion. *IRE Nat. Conv. Rec.*, pages 142–163, 1959.

[54] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Tr. on Signal Processing*, 41(12):3445–3462, Dec 1993.

[55] Y. Shoham and A. Gersho. Efficient bit allocation for an arbitrary set of quantizers. *IEEE Trans. Acoust., Speech, and Signal Proc.*, 36(9):1445–1453, September 1988.

[56] N. D. Sidiropoulos. Fast digital locally monotonic regression. *IEEE Tr. SP*, 45(2):389–395, Feb 1997.

[57] G. Strang and T. Nguyen. *Wavelets and Filter Banks.* Wellesley-Cambridge Press, Wellesley, MA, USA, 1996.

[58] T. E. Tremain. The government standard linear predictive coding algorithm: LPC-10. *Speech Technology*, pages 40-49, April 1982.

[59] A.V. Trushkin. Bit number distribution upon quantization of a random variable. *Probl. Inf. Trans.*, 16:76–79, 1980.

[60] International Telecommunication Union. *ITU-T Reccommendation T.82.* ITU, Geneva, Switzerland, March 1993.

[61] M. Vetterli and J. Kovačević. *Wavelets and Subband Coding.* Prentice Hall, Englewood Cliffs, NJ, USA, 1995.

[62] B. Widrow and S.D. Stearns. *Adaptive Signal Processing.* Prentice Hall, Englewood Cliffs, NJ, 1985.

[63] F. M. J. Willems. Coding for a binary independent piecewise-identically-distributed source. *IEEE Trans. on IT*, 42(6):2210–2217, November 1996.

Av. De Morges, 16
1004 Lausanne
Switzerland

+41 21 693 56 29 (work)
+41 21 693 43 12 (fax)
+41 21 625 71 40 (home)
prandoni@de.epfl.ch
http://lcavwww.epfl.ch/~prandoni

Paolo Prandoni

Experience

1996–1999 Federal Institute of Technology Lausanne, Switzerland
Research Assistant

§ Nonstationary signal processing.
§ Speech and Audio coding; Data Hiding.
§ Matlab and Java-based DSP educational tools.

1997-1998 C-Cube Microsystems Milpitas, CA, USA
DSP Software Engineer

§ Development of real-time MPEG-Audio compression algorithms.
§ Host-based solutions for Pentium architectures.
§ Custom DSP chip solutions for DVD-RAM systems.

1995-1996 Phylon Inc. Fremont, CA, USA
DSP Software Engineer

§ Development of host-based, real-time modem platforms.
§ Pentium-based, fully functional V32 and V34 modem design.

Education

1999 Ecole Polytechnique Federale de Lausanne, Switzerland

§ PhD in Electrical Engineering.
§ PhD Thesis: "Optimal Segmentation Techniques for Piecewise Stationary Signals".

1995 University of California, Berkeley

§ One-year independent graduate studies.
§ Recipient of the Education Abroad (UCB) and the Graduate Studies (Padua) Scholarships

1994 University of Padua, Italy

§ "Laurea" (MS) in Electrical Engineering.
§ Dissertation work in Computer Music.

Interests

Organic Architecture, piano plunking, non-fat-free cooking, long-haul flights.

List of Publications

P. Prandoni, M Vetterli, *Approximation and Compression of Piecewise Smooth Functions*, Philosophical Transactions Royal Society of London, to appear.

P. Prandoni, M Vetterli, *R/D Optimal Linear Prediction*, IEEE Trans. Acoustic, Speech, and Signal Processing, to appear.

P. Prandoni, M. Vetterli, *Optimal Bit Allocation with Side Information*, Proc. ICASSP99, Phoenix, AZ, USA, March 1999.

P. Prandoni, M. Vetterli, *R/D Optimal Data Hiding*, proc. "Security and Watermarking of Multimedia Contents", IS&T/SPIE Electronic Imaging Conference, San Jose, CA, Jan. 1999.

P. Prandoni, M. Vetterli, *An FIR Cascade Structure for Adaptive Linear Prediction*, IEEE Trans. Signal Processing, vol. 46, no. 9, Sept. 1998, pp. 2566-2571.

P. Prandoni, M. Vetterli, *Perceptually Hidden Data Transmission over Audio Signals*, Proc. ICASSP98, Seattle, USA, May 1998.

P. Prandoni, *Information Theory in Modem Practice*, LCAV Tech. Report no. SSC/1997/015.

P. Prandoni, M. Vetterli, *A Mixed-Framework Arithmetic Coder*, Proc. PCS97, Berlin, Germany, September 1997.

P. Prandoni, M. Goodwin, M. Vetterli. *Optimal time segmentation for signal modeling and compression*. Proc ICASSP97, vol 3, pp. 2029-2032, Munich, Germany, April 1997.

G. DePoli, P. Prandoni, *Sonological Models for Timbre Characterization*, Journal of New Music Research, Vol.26, n. 2, 1997.