Compressive Sensing in Video Encoding

Brian Azizi

Department of Physics
Centre for Scientific Computing
University of Cambridge

This dissertation is submitted for the degree of

Master of Philosophy

Selwyn College August 2016

I would like to dedicate this thesis to my loving parents ...

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text. This dissertation contains less than 65,000 words including appendices, bibliography, footnotes, tables and equations and has less than 150 figures.

Brian Azizi
August 2016

Acknowledgements

And I would like to acknowledge ...

Abstract

This is where you write your abstract ...

Table of contents

List of figures **viii**

List of tables **ix**

Nomenclature **x**

1 Introduction **1**

2 Compressive Sensing **2**

2.1 Signal Acquisition and Compression 2

2.1.1 Acquisition 2

2.1.2 Compression 4

2.1.3 A Novel Approach: Compressed Sampling 6

2.2 The Compressive Sensing Problem 6

2.3 The Solution 7

2.3.1 Stable Measurement Matrix 7

2.3.2 Reconstruction Algorithms 7

3 Wavelets **8**

3.1 Discrete Cosine Transform 8

3.2 Wavelet transforms 8

3.2.1 Haar wavelets 8

3.2.2 Daubechies Wavelets 9

3.3 Forming the basis matrix 9

4 Sparse Bayesian Learning **10**

4.1 Model Specification 10

4.2 Model Inference 13

4.2.1 Original Training Algorithm 15

4.2.2 Sequential Sparse Bayesian Learning Algorithm 16
4.3 Making Predictions 19

5 Design of the Multi-Scale Cascade of Estimations Algorithm 21
5.1 Interpolator 21

6 Implementation Details and Code optimization 23
6.1 Haar basis functions 23
6.1.1 1D Haar wavelet transform 23
6.1.2 2D Haar wavelet transform 24
6.1.3 3D Haar wavelet transform 26
6.2 Update formulae, details on RVM implementation 26

7 Results 27

8 Conclusion 29

References 30

List of figures

2.1 Illustration of Nyquist Sampling 3
2.2 Image Compression using DCT 5

3.1 Original image $\boldsymbol{x}$ (left) and its Haar basis transformation $\boldsymbol{w}$ (right). See next chapter for more details on Haar wavelets. 9

4.1 Plots of the Marginal Likelihood Function 17

5.1 Corrupted signal $\boldsymbol{y}$ (left) and reconstructed signal $\hat{\boldsymbol{x}}$ (right) using a cascade of 3 RVMs with Haar basis functions (see [3]). 22

7.1 Sample frame from corrupted video (left) and the reconstructed video (right) 28
7.2 Sample frame from corrupted video (left) and the reconstructed video (right) 28

List of tables

Nomenclature

Roman Symbols

M	Number of basis functions
N	Number of training examples
$\boldsymbol{w}$	RVM weights vector
$\boldsymbol{x}^{(i)}$	ith input vector
$y^{(i)}$	ith target

Greek Symbols

$\boldsymbol{\phi}_j$	jth basis vector
$\phi_j(\cdot)$	jth basis function

Chapter 1

Introduction

There are three parts: A signal processing framework called *Compressive Sensing (CS)*, a pre-processing step in form of a basis transformation based on discrete wavelet transforms and a Machine Learning algorithm called *Sparse Bayesian Learning*.

The key notion that ties in these three areas is the notion of *sparsity*.

Background

Chapter 2

Compressive Sensing

In this chapter we will outline the main ideas of the *Compressive Sensing* framework (also known as *Compressed Sensing, Compressed Sampling, Compressive Samping* or *CS*). We begin by discussing conventional approaches to data acquisition and compression, and how CS differs from that. Next, formulate the CS Problem. Following that, we briefly discuss the general solution strategy. Lastly, give a brief overview of deterministic approaches to CS reconstruction (maybe).

In Chapter 5, we will discuss a Bayesian approach to solving the Compressive Sensing Problem.

2.1 Signal Acquisition and Compression

2.1.1 Acquisition

In order to work with information within analog signals (continuous streams of data) such as sounds, images or video, we rely on reducing the analog signals to digital (discrete) signals that can be processed with computers. This digitization is done by taking discrete measurements of the analog signal at certain points in time or space, a process known as *sampling.*

Conventional approaches to sampling are based on the *Shannon/Nyquist Sampling Theorem* [4]: When sampling a band-limited signal uniformly, we are able to *perfectly reconstruct* the signal from its samples if the sampling rate is at least twice the bandwidth of the signal.

We briefly illustrate this in Figure 2.1. Consider an analog signal $x(t)$ that varies with time, such as an audio wave. Let f be the highest frequency present in $x(t)$.

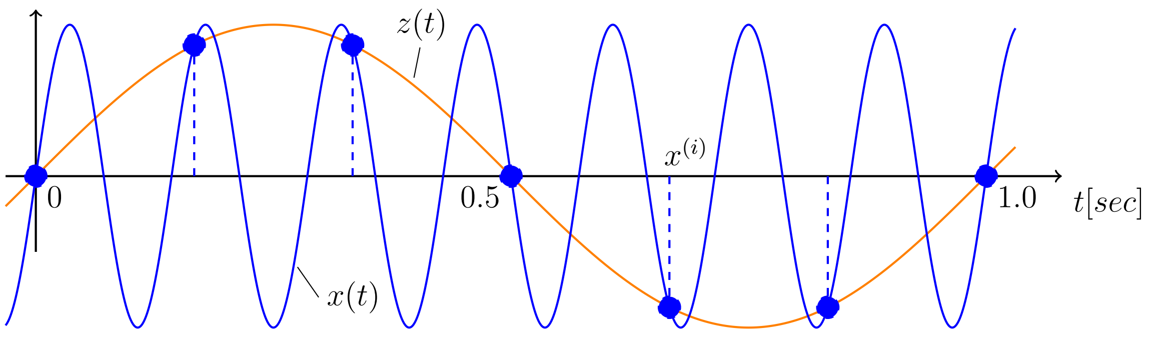


Fig. 2.1 Illustration of the Shannon/Nyquist Sampling Theorem. The orange curve is the original signal $x(t)$ which is a sinusoid with frequency $f = 7$ Hz. The blue points are discrete samples $x^{(i)}$ taken from $x(t)$ at a sampling rate $f_s = 7$ Hz, which is below the Nyquist Rate $2f = 14$ Hz. Thus, aliasing occurs and interpolation algorithms will reconstruct an alias $z(t)$ of $x(t)$.

In order to digitise $x(t)$, we measure x at discrete points in time $t^{(0)}, \cdots, t^{(n)}$ and store the samples $x^{(i)} \equiv x(t^{(i)})$. We sample x uniformly, measuring a sample every T_s seconds, so that $t^{(i)} = iT_s$. The sampling rate is therefore $f_s = 1/T_s$.

Suppose we wish to reconstruct $x(t)$ by interpolating the samples. There is an infinite number of continuous functions that fit this set of samples. However, it can be shown that only one of them has a bandwidth of no more than $f_s/2$. Thus, if $f < f_s/2$ (the *Nyquist Criterion*), then $x(t)$ is the unique function that will be approximated by interpolation algorithm such as the *Whittaker-Shannon interpolation formula*[4].

In Figure 2.1, we have a sinusoidal signal $x(t)$ with frequency f. The sampling rate is $f_s = f$ and therefore below the signal's *Nyquist rate* $2f$. Thus, we are unable to reconstruct $x(t)$ from the samples. Instead, we will reconstruct an *alias* $z(t)$ which, in this case, is another sinusoid with frequency $f/7$. The original signal x is lost.

We have illustrated Nyquist sampling in the 1-dimensional case. The same principles hold for higher dimensional signals such as images and videos.

For signals that vary with space, the sampling rate is governed by the desired spatial resolution. In order to recover the finer details (the high-frequency components) of an image, we require higher pixel density (i.e. more pixels per

Nyquist sampling underlies almost all signal acquisition protocols that are found in practice. It is the basis of medical imaging devices, consumer electronics such as audio and video recorders, radio receivers, etc.

2.1.2 Compression

The sampling theorem imposes a lower bound on the sampling rate above which we get are able to perfectly reconstruct the desired signal. This lower bound is often very high and we end up with a very large number of measurements. Storage and transfer of such signals becomes prohibitavely expensive as the size of the signal grows. Thus, a need for *data compression* arises.

We will discuss a particular type data compression known as *transform coding*. It is the standard compression method for "natural" and manmade signals such as audio, photos, and video and is the basis of many common signal formats such as JPEG for images, MPEG for videos and MP3 for audio.

Let $\boldsymbol{v}$ by a real-valued digital signal of length M, $\boldsymbol{v} \in \mathbb{R}^M$. Without loss of generality, $\boldsymbol{v}$ is assumed to be a one-dimensional signals. If we are working with a multi-dimensional signals, we may first vectorize it into a long vector. When compressing digital signals, we are usually interested in *lossy compression*.

Any vector in $\mathbb{R}^M$ can be expressed as a linear combination of M *basis vectors* $\boldsymbol{\psi}_j \in \mathbb{R}^M$:

$$\boldsymbol{v} = \sum_{j=1}^{M} w_j \boldsymbol{\psi}_j \tag{2.1}$$

where w_j is the coefficient (or weight) associated with $\boldsymbol{\psi}_j$.

By forming the *basis matrix* $\boldsymbol{\Psi} = [\boldsymbol{\psi}_1 \cdots \boldsymbol{\psi}_M]$, we can express equation (2.1) in matrix form

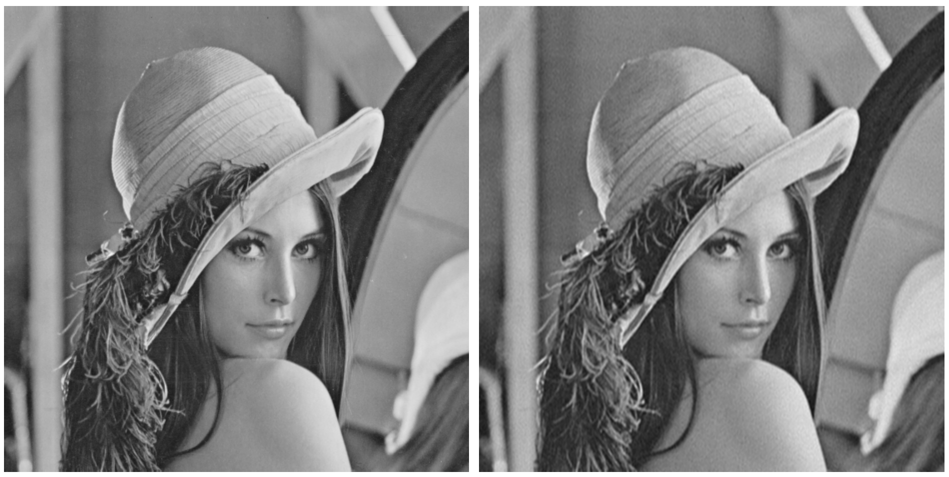
$$\boldsymbol{v} = \boldsymbol{\Psi}\boldsymbol{w}$$

where $\boldsymbol{w} = (w_1, \cdots, w_M)^T$. For simplicity, we assume that the basis $\boldsymbol{\Psi}$ is orthonormal, so that $\boldsymbol{\Psi}\boldsymbol{\Psi}^T = \boldsymbol{I}_M$ and $\boldsymbol{\psi}_i^T \boldsymbol{\psi}_j$ is 1 if $i = j$ and 0 otherwise. Thus, the coefficient w_j is given by $w_j = \boldsymbol{v}^T \boldsymbol{\psi}_j$.

We now have two equivalent representations of the same signal, $\boldsymbol{v}$ in the original basis and $\boldsymbol{w}$ in the $\boldsymbol{\Psi}$ basis. Since $\boldsymbol{\Psi}$ is orthogonal, $\boldsymbol{v}$ and $\boldsymbol{w}$ have the same ℓ_2-norm, $||\boldsymbol{v}||_2 = ||\boldsymbol{\Psi}\boldsymbol{w}||_2 = ||\boldsymbol{w}||_2$.

While in the original signal $\boldsymbol{v}$ the energy is spread over many of its components, it is possible to find a basis $\boldsymbol{\Psi}$ such that the energy of the transformed signal $\boldsymbol{w}$ is concentrated in only a few large components w_j. Thus a large fraction of the entries in $\boldsymbol{w}$ are very close to zero.

Suppose that we delete the entries w_j that are very small and replace them with zero to obtain $\hat{\boldsymbol{w}}$. Let $\hat{\boldsymbol{v}} = \boldsymbol{\Psi}\hat{\boldsymbol{w}}$ the approximate signal in the original domain. Since



(a) Uncompressed Image (b) Compressed Image via DCT

Fig. 2.2 The uncompressed image has a resolution of 512×512, i.e. 262144 pixels. We compress the image by performing a Discrete Cosine Transform and storing only the largest 27832 coefficients. The compression ratio is 9.42.

$\hat{\boldsymbol{w}}$ is very close to $\boldsymbol{w}$, so that $||\hat{\boldsymbol{w}} - \boldsymbol{w}||_2$ is very small, it follows that

$$||\hat{\boldsymbol{v}} - \boldsymbol{v}||_2 = ||\Psi\hat{\boldsymbol{w}} - \Psi\boldsymbol{w}||_2 = ||\Psi(\hat{\boldsymbol{w}} - \boldsymbol{w}||_2 = ||\hat{\boldsymbol{w}} - \boldsymbol{w})||_2$$

is also very small.

Thus, a viable method for lossy compression of the signal $\boldsymbol{v} \in \mathbb{R}^M$ would be the following:

1. Compute the full set of transform coefficients $\{w_j\}_{j=1}^M$ via $\boldsymbol{w} = \Psi^T\boldsymbol{v}$.
2. Locate all the coefficients w_j whose absolute value is above a certain threshold (suppose there are K of them).
3. Discard all the $(M - K)$ small coefficients
4. Store the values and locations of the K large coefficients

It is possible to find basis matrices $\boldsymbol{\Psi}$ that result in very high compression ratios for a wide range of natural signals without any noticable reduction in the signal quality. Furthermore, many of the commonly used basis transforms can be computed very efficiently.

Audio signals and a wide class of communication signals are highly compressible in the localized Fourier basis. Images and video signals are often compressed via the *Discrete Cosine Transform* (DCT) or the *Discrete Wavelet Transform* (DWT). For instance, the JPEG standard for image compression is based on the DCT, while the more modern JPEG2000 format uses the CDF 9/7 wavelet transform or the CDF 5/3 wavelet transform [6].

In Figure 2.2, we compress the standard test image "Lenna" via a DCT. We are only storing about 10% of the transform coefficients. Yet, the difference between the original image and the compressed image is hardly noticable.

We will discuss the DCT and the DWT in more detail in Chapter 3.

2.1.3 A Novel Approach: Compressed Sampling

Circumvent Shannon by non-uniform sampling or by using alternative sampling matrices.

2.2 The Compressive Sensing Problem

This section is based on [3]. The problem to be solved can be formulated as follows: Let $\boldsymbol{x} \in \mathbb{R}^N$ be a signal of interest. We do not measure $\boldsymbol{x}$ directly and it is thus unknown. Instead, we have a measurement $\boldsymbol{y} \in \mathbb{R}^M$, with $M << N$, from which we want to reconstruct $\boldsymbol{x}$. The signals $\boldsymbol{x}$ and $\boldsymbol{y}$ are related as follows:

$$\boldsymbol{\Omega x} = \boldsymbol{y} \tag{2.2}$$

where $\boldsymbol{\Omega}$ is a known $M \times N$ matrix referred to as the *sensing matrix*.

For example, in [3], the signal of interest $\boldsymbol{x}$ is an image, so that N is equal to the total number of pixels in the image and x_i is equal to the intensity of the corresponding pixel. However, we imagine that we have only access to a corrupted version of $\boldsymbol{x}$ in which random pixel values have been deleted. This is our measurement $\boldsymbol{y}$. The sensing matrix $\boldsymbol{\Omega}$ corresponding to this scenario is obtained by taking the $N \times N$ identity matrix and deleting the rows that correspond to the missing entries in $\boldsymbol{x}$.

Compressive Sensing (CS) is a collection of signal processing techniques that allow for efficient *reconstruction* (and indeed *aquisition*) of such signals by solving the underdetermined system (2.2).

Of course, there are infinitely many solutions to an underdetermined system. In the CS framework, we seek to find a solution $\hat{\boldsymbol{x}}$ that is *sparsest in some domain*. By that, we mean that we want to find $\hat{\boldsymbol{x}}$ that satisfies (2.2), such that there exists a basis transformation of $\hat{\boldsymbol{x}}$ in which it has the smallest number of nonzero entries.

More concretely, we assume there exists a domain in which the desired signal $\boldsymbol{x}$ is sparse. I.e. there exists a $N \times N$ basis matrix Ψ such that $\boldsymbol{x} = \boldsymbol{\Psi w}$ and $\boldsymbol{w}$ is sparse.

The CS problem can then be expressed as follows:

$$\min ||\boldsymbol{w}||_0 \qquad \text{subject to} \qquad \boldsymbol{\Omega\Psi w} = \boldsymbol{y} \tag{2.3}$$

where $||.||$ denotes the l_0 norm, i.e. the number of nonzero components.

2.3 The Solution

2.3.1 Stable Measurement Matrix

Matrices

2.3.2 Reconstruction Algorithms

L0, L1, L2, Deterministic, Geometry

For a more detailed review of the CS framework, see [1].

Chapter 3

Wavelets

In this chapter, we will introduce wavelet functions and the *Discrete Wavelet Transform* (DWT). The DWT allows us to transform our signal into a new basis in which it is sparse.

$$\boldsymbol{x} = \boldsymbol{\Psi}\boldsymbol{w} \tag{3.1}$$

that takes the dense signal $\boldsymbol{x}$ and sends it into a domain in which its transformation $\boldsymbol{w} = \boldsymbol{\Psi}^T\boldsymbol{x}$ is sparse.

3.1 Discrete Cosine Transform

An aside on the DCT. Used in old JPEG standard (ref). Formulae. Interpretations. Pictures. However, before we discuss wavelets, we will briefly introduce the *Discrete Cosine Transform* (DCT)

3.2 Wavelet transforms

3.2.1 Haar wavelets

Finding a set of basis functions $\boldsymbol{\Psi}$ that achieve such a transformation lies at the heart of many lossy compression techniques. For instance, in image processing the JPEG 2000 standard is a widely used lossy compression technique that relies on this principle. The original image $\boldsymbol{x}$ is transformed into $\boldsymbol{w}$ using the so-called *Discrete Cosine Transform.* The basis matrix $\boldsymbol{\Psi}$ is orthogonal, so $\boldsymbol{x}$ and $\boldsymbol{w}$ have the same l_2 norm. In the original signal $\boldsymbol{x}$ the length is spread across many of its coefficients. On the other hand, most of the length of $\boldsymbol{w}$ is concentrated in a few of its coefficients. A large fraction of the

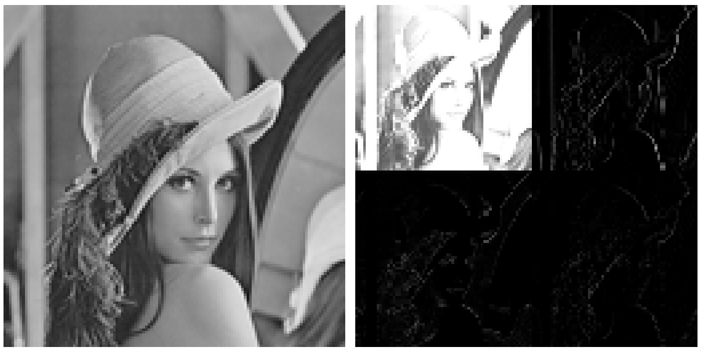


Fig. 3.1 Original image $\boldsymbol{x}$ (left) and its Haar basis transformation $\boldsymbol{w}$ (right). See next chapter for more details on Haar wavelets.

entries in $\boldsymbol{w}$ are very close to zero. By deleting these entries in $\boldsymbol{w}$ and only storing the non-zero coefficients (and the corresponding basis functions), we can obtain a compressed version $\hat{\boldsymbol{w}}$. This allows us to significantly reduce the amount of data that needs to be stored without affecting the visual quality in the reconstructed image $\hat{\boldsymbol{x}} = \boldsymbol{\Psi}\hat{\boldsymbol{w}}$.

It is important to note here that the choice of basis functions $\boldsymbol{\Psi}$ typically has a significant effect on the performance of the reconstruction algorithms.

The simplest wavelet basis transformation is based on the Haar wavelets. Figure 3.1 vizualises a basis transformation of the original image $\boldsymbol{x}$ to $\boldsymbol{w}$. This example uses a Haar wavelet basis at the first scale. We will explain the Haar wavelet transformations of images and videos in more detail in the next chapter. Dark areas correspond to small coefficients. Note that most entries in $\boldsymbol{w}$ are near zero. In practice, we approximate these entries as zero and treat $\boldsymbol{w}$ as sparse.

3.2.2 Daubechies Wavelets

Intuition. Where do the coeffs come from. Matrices. Boundary conditions.

3.3 Forming the basis matrix

1D, 2D, 3D case. Different scales

For a deeper introduction into wavelets see [5]. For more information on wavelet compression techniques, see [2].

Chapter 4

Sparse Bayesian Learning

Sparse Bayesian Learning [8] is a general Bayesian framework within supervised Machine Learning. It can be applied to both regression and classification tasks. The *Relevance Vector Machine*, or *RVM*, is a particular specialisation of the Sparse Bayesian Learning model which has identical functional form to the Support Vector Machine (SVM). However, the RVM comes with a number of key advantages over the SVM. The solution produced by a RVM is typically much sparser than the solution by a comparable SVM. Furthermore, the RVM is a probabilistic model and as such, allows us to estimate error bounds in its predictions.

In this chapter, we will derive the Sparse Bayesian Learning model for regression. We will summarise both the original inference algorithm [8] and also the faster "Sequential Sparse Bayesian Learning Algorithm" [9].

4.1 Model Specification

We are given a data set of N input vectors $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$ and their associated *targets* $\{y^{(i)}\}_{i=1}^{N}$. The input vectors live in D-dimensional space, $\boldsymbol{x} \in \mathbb{R}^D$. The targets are real values, $y \in \mathbb{R}$. [1]

We model the data using a linearly-weighted sum of M fixed basis functions $\{\phi_j(\cdot)\}_{j=1}^{M}$ and base our predictions on the function $f(\cdot)$ defined as

$$f(\boldsymbol{x}; \boldsymbol{w}) = \sum_{j=1}^{M} w_j \phi_j(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) \tag{4.1}$$

[1] When using the Sparse Bayesian model for regression, we assume the targets are real-valued. It is also possible to use the model for classification in which cased the targets are assumed to be discrete class labels.

where $\boldsymbol{w} = [w_1, \ldots, w_M]^T$ and $\boldsymbol{\phi}(\cdot) = [\phi_1(\cdot), \ldots, \phi_M(\cdot)]^T$. Using a large number M of non-linear basis functions $\phi_j : \mathbb{R}^D \to \mathbb{R}$ allows for a highly flexible model.

The *Relevance Vector Machine*, or RVM, is a specialisation of the Sparse Bayesian Learning model in which the basis functions take the form of *kernel functions*

$$\phi_j(\cdot) \equiv K\left(\cdot\,,\, \boldsymbol{x}^{(j)}\right).$$

This defines a basis function for each training data point $\boldsymbol{x}^{(i)}$. Typically, we also include an additional *bias* term $\phi_0(\cdot) \equiv 1$, so that $M = N + 1$. The RVM has identical functional form to the popular Support Vector Machine (SVM), but superior properties. It typically gives sparser solutions than the SVM and has the additional advantage of providing confidence measures for its predictions.

However, in the following derivation, we will stick to the case of general basis functions $\phi_j : \mathbb{R}^D \to \mathbb{R}$. Thus M need not equal $N+1$ and may, in fact, be a lot larger.

To train the model (4.1), i.e. find values for $\boldsymbol{w}$ that are optimal in some sense, we make the standard assumption that our training data are samples from the model with additive noise:

$$\begin{aligned} y^{(i)} &= f(\boldsymbol{x}^{(i)}; \boldsymbol{w}) + \epsilon^{(i)} \\ &= \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}^{(i)}) + \epsilon^{(i)} \qquad i = 1, \ldots, N. \end{aligned} \tag{4.2}$$

The errors $\{\epsilon^{(i)}\}_{i=1}^N$ are assumed to be independent samples from a zero-mean Gaussian distribution with variance σ^2

$$p(\epsilon^{(i)}) = \mathcal{N}(\epsilon^{(i)} \,|\, 0, \sigma^2) \qquad i = 1, \ldots, N. \tag{4.3}$$

Combining equation (4.2) with equation (4.1), we may express the model for the complete data using matrix notation:

$$\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{w} + \boldsymbol{\epsilon} \tag{4.4}$$

where $\boldsymbol{\epsilon} = \left[\epsilon^{(1)}, \cdots, \epsilon^{(N)}\right]^T$. The $N \times M$ matrix $\boldsymbol{\Phi}$ is known as the design matrix. The ith row of $\boldsymbol{\Phi}$ is given by $\boldsymbol{\phi}(\boldsymbol{x}^{(i)})^T$. The jth column of $\boldsymbol{\Phi}$ is given by $\boldsymbol{\phi}_j = \left[\phi_j\left(\boldsymbol{x}^{(1)}\right), \cdots, \phi_j\left(\boldsymbol{x}^{(N)}\right)\right]^T$, which is also referred to as the jth *basis vector*. Thus

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\phi}_1 & \cdots & \boldsymbol{\phi}_M \end{bmatrix} = \begin{bmatrix} \boldsymbol{\phi}(\boldsymbol{x}^{(1)})^T \\ \vdots \\ \boldsymbol{\phi}(\boldsymbol{x}^{(N)})^T \end{bmatrix}$$

Combining equation (4.4) and equation (4.3), we find that the complete data likelihood function is given by

$$\begin{aligned} p\left(\boldsymbol{y} \mid \boldsymbol{w}, \sigma^2\right) &= \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{w}, \sigma^2 \boldsymbol{I}_M\right) \\ &= (2\pi\sigma^2)^{-N/2} \exp\left\{-\frac{1}{2\sigma^2}||\boldsymbol{y} - \boldsymbol{\Phi w}||^2\right\} \end{aligned} \tag{4.5}$$

where $\boldsymbol{I}_M$ is the $M \times M$ identity matrix.

So far, we have specified the general linear regression model. To get to the sparse Bayesian formulation, we define a zero-mean Gaussian prior distribution over the parameters $\boldsymbol{w}$

$$p(\boldsymbol{w} \mid \boldsymbol{\alpha}) = \prod_{j=1}^{M} \mathcal{N}\left(w_j \mid \alpha_j^{-1}\right)$$

where $\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_M]^T$ is a vector of M hyperparameters. It is important to note that each hyperparameter α_j is solely responsible for controlling the strength of the prior of its associated weight w_j. If α_j is large, the prior over w_j is very strongly peaked at zero. This form of the prior distribution is, more than anything, responsible for the dramatic sparsity in the final model.

To complete the specification, we must define a prior over the noise parameter σ^2 and the a hyperprior over the hyperparameters $\boldsymbol{\alpha}$. Following the derivation in [8], we use the following Gamma [2] priors

$$p(\boldsymbol{\alpha} \mid a, b) = \prod_{j=1}^{M} \text{Gamma}(\alpha_j \mid a, b)$$

$$p(\beta \mid c, d) = \text{Gamma}(\beta \mid c, d)$$

where $\beta \equiv \sigma^{-2}$.

As a side note, consider the prior of $\boldsymbol{w}$ after marginalising out the dependence on the hyperpriors $\boldsymbol{\alpha}$. Since each w_j is normally distributed with an unknown precision parameter α_j and since the (hyper)prior over α_j is the Gamma distribution and

[2] The Gamma distribution is defined by

$$\text{Gamma}(z \mid a, b) = \Gamma(a)^{-1} b^a z^{a-1} \exp(-bz) \qquad z, a, b > 0$$

where $\Gamma(.)$ is the Gamma function defined by

$$\Gamma(z) = \int_0^\infty t^{z-1} \exp(-t) dt.$$

therefore conjugate to $p(w_j \,|\, \alpha_j)$, it follows that the resulting integral can be evaluated analytically

$$\begin{aligned} p(\boldsymbol{w} \,|\, a, b) &= \int p(\boldsymbol{w} \,|\, \boldsymbol{\alpha}) p(\boldsymbol{\alpha} \,|\, a, b) d\boldsymbol{\alpha} \\ &= \prod_{j=1}^{M} \int \mathcal{N}(w_j \,|\, 0, \alpha_j^{-1}) \, \mathrm{Gamma}(\alpha_j | \, a, b) d\alpha_j \\ &= \prod_{j=1}^{M} \frac{b^a \Gamma(a + \frac{1}{2})}{(2\pi)^{\frac{1}{2}} \Gamma(a)} \left(b + \frac{w_j^2}{2} \right)^{-(a+\frac{1}{2})} . \end{aligned}$$

This corresponds to a product of independent Student-t density functions over the weights w_j. The choice $a = b = 0$ implies that $p(\boldsymbol{w} \,|\, a, b) \propto \prod_{j=1}^{M} 1/|w_j|$. As discussed in [8], it is this hierarchical formulation of the weight prior that is ultimately responsible for encouraging sparse solutions.

4.2 Model Inference

We have specified the likelihood model for the data and a prior distribution over the model parameters. The next step in Bayesian inference is to compute the posterior distribution of the parameters. We begin by setting up Bayes' Rule

$$p(\boldsymbol{w}, \boldsymbol{\alpha}, \sigma^2 \,|\, \boldsymbol{y}) = \frac{p(\boldsymbol{y} \,|\, \boldsymbol{w}, \boldsymbol{\alpha}, \sigma^2) p(\boldsymbol{w}, \boldsymbol{\alpha}, \sigma^2)}{\int p(\boldsymbol{y} \,|\, \boldsymbol{w}, \boldsymbol{\alpha}, \sigma^2) p(\boldsymbol{w}, \boldsymbol{\alpha}, \sigma^2) d\boldsymbol{w} d\boldsymbol{\alpha} d\sigma^2} \tag{4.6}$$

The integral in the denominator of (4.6) is computationally intractable and we must resort to an alternative strategy. First, we decompose the left-hand-side of equation (4.6) as

$$p(\boldsymbol{w}, \boldsymbol{\alpha}, \sigma^2 \,|\, \boldsymbol{y}) = p(\boldsymbol{w} \,|\, \boldsymbol{y}, \boldsymbol{\alpha}, \sigma^2) p(\boldsymbol{\alpha}, \sigma^2 \,|\, \boldsymbol{y}).$$

Next, we use Bayes' Rule to compute the posterior distribution of the weights given $\boldsymbol{\alpha}$ and σ^2

$$p(\boldsymbol{w} \,|\, \boldsymbol{y}, \boldsymbol{\alpha}, \sigma^2) = \frac{p(\boldsymbol{y} \,|\, \boldsymbol{w}, \sigma^2) p(\boldsymbol{w} \,|\, \boldsymbol{\alpha})}{p(\boldsymbol{y} \,|\, \boldsymbol{\alpha}, \sigma^2)} \tag{4.7}$$

The denominator of the right-hand-side is known as the *marginal likelihood* and given by

$$p(\boldsymbol{y} \,|\, \boldsymbol{\alpha}, \sigma^2) = \int p(\boldsymbol{y} \,|\, \boldsymbol{w}, \sigma^2) p(\boldsymbol{w} \,|\, \boldsymbol{\alpha}) d\boldsymbol{w} \tag{4.8}$$

Since $\boldsymbol{\alpha}$ and σ^2 are treated as fixed quantities in equation (4.7), the Gaussian density $p(\boldsymbol{w} \,|\, \boldsymbol{\alpha})$ is the conjugate prior to the Gaussian likelihood function $p(\boldsymbol{y} \,|\, \boldsymbol{w}, \sigma^2)$. Thus, the integral in equation (4.8) is a convolution of two Gaussians and therefore equal to another Gaussian:

$$\begin{aligned} p(\boldsymbol{y}\,|\,\boldsymbol{\alpha},\sigma^2) &= \mathcal{N}(\boldsymbol{y}\,|\,\boldsymbol{0},\boldsymbol{C}) \\ &= (2\pi)^{-N/2}|\boldsymbol{C}|^{-1/2}\exp\left\{-\frac{1}{2}\boldsymbol{y}^T\boldsymbol{C}^{-1}\boldsymbol{y}\right\} \end{aligned}$$

where

$$\boldsymbol{C} = \sigma^2\boldsymbol{I}_N + \boldsymbol{\Phi}\boldsymbol{A}^{-1}\boldsymbol{\Phi}^T. \tag{4.9}$$

The posterior distribution for $\boldsymbol{w}$ is a also a Gaussian:

$$p(\boldsymbol{w}\,|\,\boldsymbol{y},\boldsymbol{\alpha},\sigma^2) = \mathcal{N}(\boldsymbol{w}\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma}). \tag{4.10}$$

Its mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ are given by

$$\boldsymbol{\Sigma} = \left(\sigma^{-2}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{A}\right)^{-1} \tag{4.11}$$

$$\boldsymbol{\mu} = \sigma^{-2}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{y} \tag{4.12}$$

with $\boldsymbol{A} = \mathrm{diag}(\boldsymbol{\alpha})$.

Finally, we need to find the posterior of the hyperparameters, $p(\boldsymbol{\alpha},\sigma^2\,|\,\boldsymbol{y})$. This part is computationally intractable, so instead we approximate the posterior by a delta-function at its mode. Hence, the problem reduces to finding the values of $\boldsymbol{\alpha}$ and σ^2 that maximise $p(\boldsymbol{\alpha},\sigma^2\,|\,\boldsymbol{y}) \propto p(\boldsymbol{y}\,|\,\boldsymbol{\alpha},\sigma^2)p(\boldsymbol{\alpha})p(\sigma^2)$.

Here, we make the simplifying assumption that $a = b = c = d = 0$, giving us uniform (but improper) hyperpriors (see [8] for the general case). Maximising $p(\boldsymbol{\alpha},\sigma^2\,|\,\boldsymbol{y})$ is then equivalent to maximising the marginal likelihood, or equivalently, its logarithm

$$\begin{aligned} \mathcal{L}(\boldsymbol{\alpha},\sigma^2) &= \log p(\boldsymbol{y}\,|\,\boldsymbol{\alpha},\sigma^2) = \log\mathcal{N}(\boldsymbol{y}\,|\,\boldsymbol{0},\boldsymbol{C}) \\ &= -\frac{1}{2}\left[N\log 2\pi + \log|\boldsymbol{C}| + \boldsymbol{y}^T\boldsymbol{C}^{-1}\boldsymbol{y}\right] \end{aligned} \tag{4.13}$$

The procedure of finding $\boldsymbol{\alpha}$ and σ^2 that maximise the (log) marginal likelihood (4.13) is also known as *type-II Maximum likelihood* and *evidence approximation.*

Algorithm 1 Sparse Bayesian Learning: Original Training Algorithm

```
 1: Choose some initial positive values for $\sigma^2$ and $\alpha_j$ for $j = 1, \cdots, M$
 2: repeat
 3:     $\boldsymbol{A} = \mathrm{diag}(\boldsymbol{\alpha})$
 4:     $\boldsymbol{\Sigma} = \left(\sigma^{-2}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{A}\right)^{-1}$
 5:     $\boldsymbol{\mu} = \sigma^{-2}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{y}$

 6:     for $j = 1, \cdots, M$ do
 7:         $\gamma_j = 1 - \alpha_j\Sigma_{jj}$
 8:         $\alpha_j = \gamma_j/\mu_j^2$
 9:     end for
10:     $\sigma^2 = ||\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\mu}||^2/(N - \sum_j \gamma_j)$
11: until Convergence
```

4.2.1 Original Training Algorithm

The original training algorithm in [8] is derived by setting the derivatives of (4.13) to zero. We obtain the following update equations for $\boldsymbol{\alpha}$ and σ^2:

$$\alpha_j^{\text{new}} = \frac{\gamma_j}{\mu_j^2} \tag{4.14}$$

$$(\sigma^2)^{\text{new}} = \frac{||\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\mu}||^2}{N - \sum_j \gamma_j} \tag{4.15}$$

where μ_j is the jth component of the posterior mean $\boldsymbol{\mu}$ (4.12). The quantities γ_j are defined by

$$\gamma_j = 1 - \alpha_j\Sigma_{jj}$$

where Σ_{jj} is the jth diagonal element of the posterior covariance $\boldsymbol{\Sigma}$ (4.11).

To train the model, we can start by giving $\boldsymbol{\alpha}$ and σ^2 some initial values and evaluate the mean and covariance of the weights posterior using equations (4.12) and (4.11), respectively. Next we alternate between re-estimating the hyperparamaters $\boldsymbol{\alpha}$ and σ^2 using (4.14) and (4.15) and updating the posterior mean and covariance parameters using (4.12) and (4.11). We continue until a relevant convergence criterion is met. For example, we may choose to stop if the change in the marginal likelihood - or, alternatively, the change in the parameter values - between two iterations is below a certain pre-defined threshold.

This procedure is summarised in Algorithm 1.

During training, it is typically observed that many of the hyperparameters α_j tend to infinity. Equations (4.12) and (4.11) imply that the weights w_j corresponding to these hyperparameters have a posterior distribution where the mean and the variance are both zero, meaning their posterior is infinitely peaked at zero. As a consequence, the corresponding basis functions $\phi_j(\cdot)$ are effectively removed from the model and we achieve sparsity.

In the case of the RVM, where $\phi_j(\cdot) \equiv K(\cdot, \boldsymbol{x}^{(j)})$, the input vectors $\boldsymbol{x}^{(j)}$ corresponding to the remaining non-zero weights are known as the *relevance vectors* of the model.

4.2.2 Sequential Sparse Bayesian Learning Algorithm

A central drawback of the training algorithm discussed in the previous section is its speed. The computational complexity scales with the cube of the number of basis functions. During training, as basis functions are pruned from the model, the algorithm accelerates. Nevertheless, if M is very large, the procedure can be very expensive to run.

An alternative strategy of maximising the marginal likelihood (4.13) was developed by [9], resulting in a highly accelerated training algorithm: the *Sequential Sparse Bayesian Learning Algorithm*. It starts with a single basis function and maximises the marginal likelihood by sequentially adding and deleting candidate basis functions. This significantly reduces the computational complexity of the algorithm.

To derive the algorithm, we follow the analysis in [7] and consider the dependence of the marginal likelihood $\mathcal{L}(\boldsymbol{\alpha}, \sigma^2)$ on a single hyperparameter α_j. First, we decompose the matrix $\boldsymbol{C}$, defined in (4.9), as follows:

$$\begin{aligned} \boldsymbol{C} &= \sigma^2 \boldsymbol{I}_N + \sum_{m \neq j} \alpha_m^{-1} \boldsymbol{\phi}_m \boldsymbol{\phi}_m^T + \alpha_j^{-1} \boldsymbol{\phi}_j \boldsymbol{\phi}_j^T \\ &= \boldsymbol{C}_{-j} + \alpha_j^{-1} \boldsymbol{\phi}_j \boldsymbol{\phi}_j^T \end{aligned}$$

where $\boldsymbol{C}_{-j} \equiv \sigma^2 \boldsymbol{I}_N + \sum_{m \neq j} \alpha_m^{-1} \boldsymbol{\phi}_m \boldsymbol{\phi}_m^T$ is $\boldsymbol{C}$ without the contribution of the jth basis vector $\boldsymbol{\phi}_j$. Making use of standard identities [WHICH ONES?] for matrix inverses and determinants, we can express $|\boldsymbol{C}|$ and $\boldsymbol{C}^{-1}$ as

$$\boldsymbol{C}^{-1} = \boldsymbol{C}_{-j}^{-1} - \frac{\boldsymbol{C}_{-j}^{-1} \boldsymbol{\phi}_j \boldsymbol{\phi}_j^T \boldsymbol{C}_{-j}^{-1}}{\alpha_j + \boldsymbol{\phi}_j^T \boldsymbol{C}_{-j}^{-1} \boldsymbol{\phi}_j}$$

$$|\boldsymbol{C}| = |\boldsymbol{C}_{-j}| \left| 1 + \alpha_j^{-1} \boldsymbol{\phi}_j^T \boldsymbol{C}_{-j}^{-1} \boldsymbol{\phi}_j \right|$$

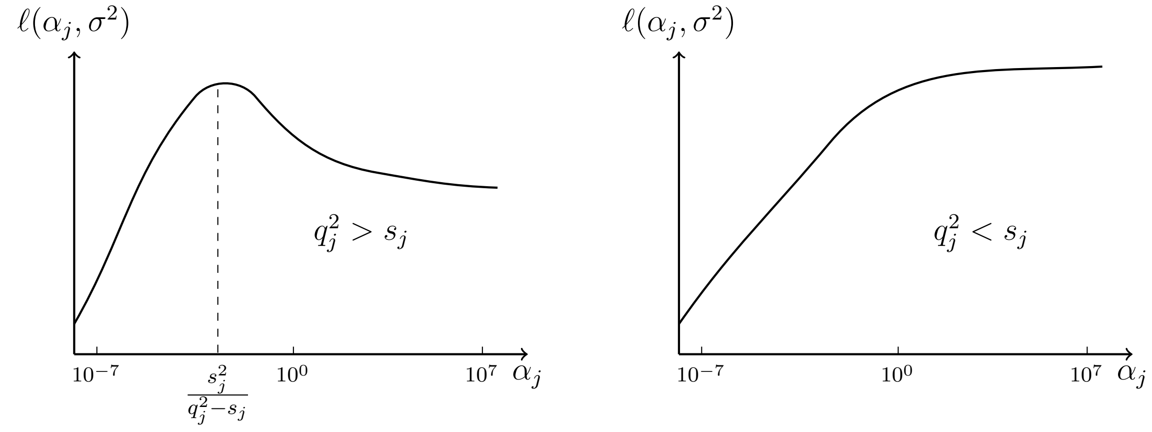


Fig. 4.1 Example plots of $\ell(\alpha_j, \sigma^2)$ against α_j illustrating the stationary points when $q_j^2 > s_j$ (left) and $q_j^2 < s_j$ (based on [7]).

This allows us to decompose the marginal likelihood:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\alpha}, \sigma^2) &= \mathcal{L}(\boldsymbol{\alpha}_{-j}, \sigma^2) + \frac{1}{2}\left[\log \alpha_j - \log(\alpha_j + s_j) + \frac{q_j^2}{\alpha_j + s_j}\right] \\ &\equiv \mathcal{L}(\boldsymbol{\alpha}_{-j}, \sigma^2) + \ell(\alpha_j, \sigma^2) \end{aligned} \tag{4.16}$$

This conveniently seperates terms in α_j in $\ell(\alpha_j, \sigma^2)$ from the remaining terms in $\mathcal{L}(\boldsymbol{\alpha}_{-j}, \sigma^2)$, which is the (log) marginal likelihood with the basis vector $\boldsymbol{\phi}_j$ excluded.

The quantity s_j is the *sparsity factor*, defined as

$$s_j = \boldsymbol{\phi}_j^T \boldsymbol{C}_{-j}^{-1} \boldsymbol{\phi}_j.$$

It serves as a measure of how much the marginal likelihood would decrease if we added $\boldsymbol{\phi}_j$ to the model. The quantity q_j, on the other hand, is known as the *quality factor.* It is defined as

$$q_j = \boldsymbol{\phi}_j^T \boldsymbol{C}_{-j}^{-1} \boldsymbol{y}$$

and measures the extent to which ϕ_j increases $\mathcal{L}(\boldsymbol{\alpha}, \sigma^2)$ by helping to explain the data $\boldsymbol{y}$. Thus, a particular basis vector $\boldsymbol{\phi}_j$ should not be included in the model if its sparsity factor s_j is large, unless it is offset by a large quality factor q_j.

We can see this more explicitly if we consider the first derivative of $\ell(\alpha_j, \sigma^2)$ with respect to α_j [7]

$$\frac{\partial \ell(\alpha_j, \sigma^2)}{\partial \alpha_j} = \frac{\alpha_j^{-1} s_j^2 - (q_j^2 - s_j)}{2(\alpha_j + s_j)^2}$$

Equating it to zero (and noting that α_j is an inverse-variance and therefore positive), we obtain the following solution for α_j:

$$\alpha_j = \begin{cases} s_j^2/(q_j^2 - s_j) & \text{if } q_j^2 > s_j \\ +\infty & \text{otherwise} \end{cases} . \tag{4.17}$$

The solution (4.17) is illustrated in Figure 4.1.

It follows that, if, during training, a candidate basis vector $\boldsymbol{\phi}_j$ is currently included in the model (meaning $\alpha_j < \infty$) even though $q_j^2 \leq s_j$, then α_j should be set to ∞ and $\boldsymbol{\phi}_j$ should be pruned from the model. On the other hand, if $\boldsymbol{\phi}_j$ is currently excluded from the model (i.e. $\alpha_j = \infty$), but $q_j^2 > s_j$, then α_j should be set to $s_j^2/(q_j^2 - s_j)$ and $\boldsymbol{\phi}_j$ should be added to the model. Furthermore, if $\boldsymbol{\phi}_j$ is included and $q_j^2 > s_j$, then we may also re-estimate α_j. Each step in the algorithm (weakly) increases the marginal likelihood. Thus we are guaranteed to find a maximum.

During the algorithm, we must maintain and update values of the quality factors and sparsity factors for all basis functions, as well as the posterior mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ of the weights $\boldsymbol{w}$. In practice, it easier to keep track of the quantities $Q_m = \boldsymbol{\phi}_m^T \boldsymbol{C}^{-1} \boldsymbol{\phi}_m$ and $S_m = \boldsymbol{\phi}_m^T \boldsymbol{C}^{-1} \boldsymbol{y}$ which can also be written as (using the Woodbury Identity)

$$S_m = \sigma^{-2} \boldsymbol{\phi}_m^T \boldsymbol{\phi}_m - \sigma^{-4} \boldsymbol{\phi}_m^T \boldsymbol{\Phi} \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \boldsymbol{\phi}_m \tag{4.18}$$

$$Q_m = \sigma^{-2} \boldsymbol{\phi}_m^T \boldsymbol{y} - \sigma^{-4} \boldsymbol{\phi}_m^T \boldsymbol{\Phi} \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \boldsymbol{y} \tag{4.19}$$

where $\boldsymbol{\Sigma}$ and $\boldsymbol{\Phi}$ contain only the basis functions that are currently included in the model.

The factors s_m and q_m can by obtained from S_m and Q_m as follows:

$$s_m = \frac{\alpha_m S_m}{\alpha_m - S_m} \tag{4.20}$$

$$q_m = \frac{\alpha_m Q_m}{\alpha_m - S_m} \tag{4.21}$$

Note that if $\alpha_m = \infty$, then $q_m = Q_m$ and $s_m = S_m$.

We have summarized the procedure in Algorithm 2. After initializing the standard deviation σ^2 in step 1, we add the first basis function $\boldsymbol{\phi}_j$ to the model. We could initialize with any basis vector, but in step 2, we pick the one with the largest normalized projection on the target vector $\boldsymbol{y}$, i.e. we choose $j = \arg\max_m \left\{ ||\boldsymbol{\phi}_m^T \boldsymbol{y}||^2 / ||\boldsymbol{\phi}_m||^2 \right\}$. In

Algorithm 2 Sequential Sparse Bayesian Learning Algorithm [9]

```
1:  Initialise $\sigma^2$.
2:  Add basis function $\boldsymbol{\phi}_j$ to the model, where $j = \arg\max_m \left\{ ||\boldsymbol{\phi}_m^T \boldsymbol{y}||^2 / ||\boldsymbol{\phi}_m||^2 \right\}$.
    Set $\alpha_j = \frac{||\phi_j||^2}{||\phi_j^T \boldsymbol{y}||^2/||\phi_j||^2 - \sigma^2}$. Set $\alpha_m = \infty$ for $m \neq j$.
3:  Compute $\boldsymbol{\Sigma} = \left(\sigma^{-2}\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{A}\right)^{-1}$ and $\boldsymbol{\mu} = \sigma^{-2}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{y}$ which are scalars initially.
    Compute $S_m$, $Q_m$, $s_m$ and $q_m$ for $m = 1, \cdots, M$ using (4.18) − (4.21).
4:  **repeat**
5:      Select some candidate basis vector $\boldsymbol{\phi}_j$.
6:      **if** $q_j^2 > s_j$ and $\alpha_j = \infty$ **then add** $\boldsymbol{\phi}_j$ to the model and update $\alpha_j$.
7:      **if** $q_j^2 > s_j$ and $\alpha_j < \infty$ **then re-estimate** $\alpha_j$.
8:      **if** $q_j^2 < s_j$ and $\alpha_j < \infty$ **then delete** $\boldsymbol{\phi}_j$ from the model and set $\alpha_j = \infty$.
9:      Update $\sigma^2 = ||\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w}||/(N - M + \sum_m \alpha_m \boldsymbol{\Sigma}_{mm})$[8].
10:     Update $\boldsymbol{\Sigma}$, $\boldsymbol{\mu}$ and $S_m$, $Q_m$, $s_m$, $q_m$ for $m = 1, \cdots, M$.
11: **until** Convergence
```

step 3 we compute the model statistics and in step 4 we begin the large loop of the algorithm. There are two things to note here. First, in step 5, we need to select a candidate basis vector $\boldsymbol{\phi}_j$. We are free to pick one at random. Alternatively, it is possible to compute the change in the marginal likelihood for each ccandidate basis vector and choose the one that would give us the largest increase. Second, we would usually like to estimate the noise variance σ^2 from the data, as is done in step 9. However, in practice, we may decide to set σ^2 in advance in step 1 and keep it fixed throughout the algorithm. If we decide to do so, then we can perform the updates in step 10 using very efficient update formulae that do not require matrix inversions. The formulae can be found in the appendix of [9]. If we do decide to update σ^2 in step 9, then we must use the full equations (4.11), (4.12) and (4.18)-(4.21).

4.3 Making Predictions

Once we have trained the model, we may use it to predict the target y^* for a new input vector $\boldsymbol{x}^*$. To do so, we would like to compute the *predictive distribution*

$$p(y^* \mid \boldsymbol{y}) = \int p(y^* \mid \boldsymbol{w}, \boldsymbol{\alpha}, \sigma^2) p(\boldsymbol{w}, \boldsymbol{\alpha}, \sigma^2 \mid \boldsymbol{y}) \, d\boldsymbol{w} \, d\boldsymbol{\alpha} \, d\sigma^2.$$

We cannot compute this integral analytically, nor do we actually know the posterior of all the model parameters. Instead, we use the type-II maximum likelihood solutions for $\boldsymbol{\alpha}$ and σ^2 that we obtained during training and base our predictions on the posterior distribution of the weights conditioned on $\boldsymbol{\alpha}$ and σ^2. The predictive distribution for $\boldsymbol{x}^*$ is then:

$$p(y^* \,|\, \boldsymbol{y}, \boldsymbol{\alpha}, \sigma^2) = \int p(y^* \,|\, \boldsymbol{w}, \sigma^2) p(\boldsymbol{w} \,|\, \boldsymbol{y}, \boldsymbol{\alpha}, \sigma^2) \, d\boldsymbol{w} \tag{4.22}$$

Both factors in the integrand are Gaussians, and we can therefore readily compute the integral to get

$$p(y^* \,|\, \boldsymbol{y}, \boldsymbol{\alpha}, \sigma^2) = \mathcal{N}(y^* \,|\, \mu^*, (\sigma^2)^*) \tag{4.23}$$

The predictive mean is given by

$$\mu^* = \boldsymbol{\mu}^T \boldsymbol{\phi}(\boldsymbol{x}^*) \tag{4.24}$$

and the predictive variance is given by

$$(\sigma^2)^* = \sigma^2 + \boldsymbol{\phi}(\boldsymbol{x}^*)^T \boldsymbol{\Sigma} \boldsymbol{\phi}(\boldsymbol{x}^*) \tag{4.25}$$

Equation (4.24) implies that, if we want to produce point predictions, we may simply set the weights $\boldsymbol{w}$ equal to posterion mean $\boldsymbol{\mu}$ which is typically very sparse. If we are also interested in error bars for our predictions, we can obtain them using Equation (4.25). The error bars consist of two parts, the noise in the data σ^2 and the uncertainty in the weights.

For more details and derivations on Sparse Bayesian Learning, see [7–9].

Chapter 5

Design of the Multi-Scale Cascade of Estimations Algorithm

Bringing all building blocks together. Description and explanation of the algorithm

1. *Bayesian CS*
2. *Basis Matrix ?*
3. *MSCE Algorithm*

So far, we have not addressed the central question: How do we solve the compressive sensing problem (2.3)? Various deterministic approaches have been developed in recent years. See [3] for an overview.

In the MPhil project, we will employ a probabilistic technique based on Sparse Bayesian Learning. In particular, we will use the *Relevance Vector Machine (RVM)* [8, 9] to reconstruct $\boldsymbol{w}$ from the measurements $\boldsymbol{y}$. Following that, we obtain a reconstructed version of the desired signal $\boldsymbol{x}$ by pre-multiplying $\boldsymbol{w}$ by $\boldsymbol{\Psi}$ to obtain the desired signal.

5.1 Interpolator

We use a sensing matrix $\boldsymbol{\Omega}$ that acts as signal mask. That is, we obtain the $N \times M$ matrix $\boldsymbol{\Omega}$ by taking the $M \times M$ identity matrix $\boldsymbol{I}_M$ and deleting $(M - N)$ rows. This corresponds to a subsampled signal in which we only measured N pixel values. For this specific class of sensing matrices, the problem of reconstructing the original signal is also known as *interpolation*.

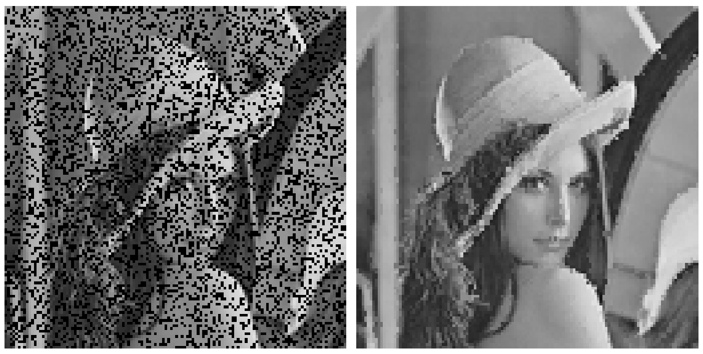


Fig. 5.1 Corrupted signal $\boldsymbol{y}$ (left) and reconstructed signal $\hat{\boldsymbol{x}}$ (right) using a cascade of 3 RVMs with Haar basis functions (see [3]).

In order to reconstruct the image, we use the estimated posterior mean to "predict" what a pixel value y^* should be at a location x^* in which information was missing:

$$y^* = \boldsymbol{w}^T \boldsymbol{\psi}(x^*) \tag{5.1}$$

Apart from achieving sparse solutions, one further desirable feature of the RVM is that the model provides error bars for its predictions. This is used in [3] to construct a multi-scale cascade of RVM estimations and achieve significant performance boosts.

An example of this can be seen in Figure 5.1.

Chapter 6

Implementation Details and Code optimization

This chapter gives a brief description of the current state of our implementation of the 3D signal reconstructer.

6.1 Haar basis functions

The RVM takes as input a target vector ($\boldsymbol{y}$) and a basis matrix ($\boldsymbol{\Psi}$). In this respect, it is agnostic about whether the signal is an image or video or of some other type alltogether. Most of this information is encoded in the basis matrix $\boldsymbol{\Psi}$. It is therefore important, and often challenging, to select a good set of basis functions.

Our current implementation uses 3-dimensional Haar wavelet basis functions. I will show how the basis matrix $\boldsymbol{\Psi}$ is constructed by briefly describing how the discrete Haar wavelet transform is performed on 1D, 2D and finally on 3D signals.

6.1.1 1D Haar wavelet transform

Consider a 1-dimensional signal $\boldsymbol{s} = \{s_1, \ldots, s_r\} \in \mathbb{R}^r$ (r for "rows"), where, for simplicity, we assume that r is a power of 2. The Haar wavelet transform can be performed at various resolution scales. The transform at the first scale is given by:

$$\boldsymbol{s} = \{s_1, \ldots, s_r\} \to \frac{1}{\sqrt{2}}\{s_1 + s_2, s_3 + s_4, \ldots, s_{r-1} + s_r, s_1 - s_2, \ldots, s_{r-1} - s_r\} = \hat{\boldsymbol{s}}^{(1)}$$

The first half of the signal is replaced by scaled averages of adjacent elements and the second half is replaced by scaled differences of adjacent elements. By performing this transform again on the first half of $\hat{\boldsymbol{s}}^{(1)}$ while keeping the second half fixed, we get the Haar wavelet transform at the second scale $\hat{\boldsymbol{s}}^{(2)}$. To get the third scale transform $\hat{\boldsymbol{s}}^{(3)}$, we perform the initial transform on the first quarter of $\hat{\boldsymbol{s}}^{(2)}$ while keeping the rest of the signal fixed. We may continue this process until we reach the ith scale, where $2^i = r$.

From here on, we will only consider the first scale transform $\hat{\boldsymbol{s}}^{(1)}$ and we will omit the (1) superscript. We can express the transform as a multiplication by an orthogonal $r \times r$ matrix W given by

$$W = \begin{bmatrix} \Phi_r \\ \Psi_r \end{bmatrix} \tag{6.1}$$

where Φ_r and Ψ_r are $(r/2) \times r$ matrices[1] given by

$$\Phi_r = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 1 \end{pmatrix}$$

and

$$\Psi_r = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & -1 \end{pmatrix}$$

In the signal processing literature, Φ_r is referred to as a low pass filter, while Ψ_r is referred to as a high pass filter. Φ_r outputs an average of the signal and Ψ_r outputs the details of the signal.

6.1.2 2D Haar wavelet transform

Let $A \in \mathbb{R}^{r \times c}$ be a 2-dimensional signal (e.g. an image). For simplicity, we will assume that both r and c are powers of 2 (though not necessarily equal).

It is simple to obtain A's Haar wavelet transform $\hat{A}$ at the first scale. This is done by first applying the 1-dimensional transform individually to each column of A to obtain a temporary matrix $\hat{A}_{temp}$. Next, we apply the 1-dimensional haar wavelet transform individually to each row of $\hat{A}_{temp}$ to obtain $\hat{A}$.

[1]Note that the matrix Ψ_r used here is different to the matrix Ψ that was used in the previous chapter (which corresponds to W^T here).

We can again express the transform as a multiplication of matrices:

$$\hat{A} = \begin{bmatrix} \Phi_r \\ \Psi_r \end{bmatrix} A \begin{bmatrix} \Phi_c^T & \Psi_c^T \end{bmatrix} \tag{6.2}$$

where Φ_r and Ψ_r are as before and Φ_c and Ψ_c are of similar form but each have dimensions $(c/2) \times c$. This is the transform that was used to generate the RHS of Figure 3.1. We note that the high-pass filters essentially detect edges of various orientations in the image.

However, as it currently stands, we cannot use this form of the basis transformation for the reconstruction algorithm. Recall that the RVM requires a *vector* of measurements as opposed to a matrix and also that it requires a single basis matrix, not a basis transform as given in (6.2).

To do this, we store the 2-dimensional signal A as a long column vector $\boldsymbol{a}$ of length rc by pasting the individual columns of A one after another. The basis transformation of $\boldsymbol{a}$ can then be expressed as

$$\hat{\boldsymbol{a}} = W\boldsymbol{a}$$

where W is a $rc \times rc$ matrix given by

$$W = \begin{bmatrix} \Phi_c \otimes \Phi_r \\ \Phi_c \otimes \Psi_r \\ \Psi_c \otimes \Phi_r \\ \Psi_c \otimes \Psi_r \end{bmatrix}$$

The symbol $\otimes$ denotes the *Kronecker product.* The kronecker product $P \otimes Q$ between matrices P and Q with dimensions $m_P \times n_P$ and $m_Q \times n_Q$, respectively, is defined to be the block matrix

$$\begin{bmatrix} p_{1,1}Q & p_{1,2}Q & \cdots & p_{1,n_P}Q \\ p_{2,1}Q & p_{2,2}Q & \cdots & p_{2,n_P}Q \\ \vdots & \vdots & \ddots & \vdots \\ p_{m_P,1}Q & p_{m_P,2}Q & \cdots & p_{m_P,n_P}Q \end{bmatrix}$$

of size $m_P m_Q \times n_P n_Q$.

6.1.3 3D Haar wavelet transform

Let $V \in \mathbb{R}^{r \times c \times s}$ be a 3-dimensional signal such as a video. V has r rows, c columns and s slices, and we assume that r, c and s are all powers of 2. We may visualize V as a "volume" with 2 spacial dimensions and one time dimension corresponding to frames of the video.

To obtain the Haar wavelet transform $\hat{V}$ of V, we first perform the 1-dimensional transform individually on each column in every slice of V to get $\hat{V}_{temp1}$. We then perform the 1D transform on every row in every slice of $\hat{V}_{temp_1}$ to get $\hat{V}_{temp_2}$. Finally, we perform the 1D transform across the slices for every row and column to get $\hat{V}$.

However, like in the 2-dimensional case, we need to be able to pass a single vector of coefficients and a single basis matrix to the RVM. To do this, we vectorize V as follows. First, we vectorize each individual slice of V as before in the 2D case. Then, we stack all these vectors on top each other to get one very long column vector $\boldsymbol{v}$ of length rcs. The Haar wavelet transform is given by

$$\hat{\boldsymbol{v}} = W\boldsymbol{v}$$

where

$$W = \begin{bmatrix} \Phi_s \otimes \Phi_c \otimes \Phi_r \\ \Phi_s \otimes \Phi_c \otimes \Psi_r \\ \Phi_s \otimes \Psi_c \otimes \Phi_r \\ \Phi_s \otimes \Psi_c \otimes \Psi_r \\ \Psi_s \otimes \Phi_c \otimes \Phi_r \\ \Psi_s \otimes \Phi_c \otimes \Psi_r \\ \Psi_s \otimes \Psi_c \otimes \Phi_r \\ \Psi_s \otimes \Psi_c \otimes \Psi_r \end{bmatrix}$$

Comparing notation to the previous chapter, what we refer to as W here is the transpose of what was previously denoted as Ψ. And since $\boldsymbol{v} = W^T\hat{\boldsymbol{v}}$, we see that $\boldsymbol{v}$ corresponds to what was previously called $\boldsymbol{x}$.

6.2 Update formulae, details on RVM implementation

Chapter 7

Results

We have obtained some results with our current implementation. The implementation uses the Haar wavelet transform at the first scale.

Our example video has a resolution of 128 by 128 pixels and consists of a total of 64 frames. Thus, $r = 128$, $c = 128$ and $s = 64$. Note that even for such a relatively small sample, the size of the basis matrix Ψ is $(128 * 128 * 64) \times (128 * 128 * 64) = 1048576 \times 1048576$. Even in single precision, storing this matrix would require around 4 terrabytes.

For this reason, we have split the original input signal into $8 \times 8 \times 8$ blocks and perform the algorithm on the individual blocks.

In Figures 3.1 and 3.2, we have included a sample frame from the corrupted test video and the same frame after reconstruction.

In Figure 3.1, we corrupted the video by deleting 30% of the pixel values in the first frame and deleting the same pixel values in each subsequent frame (so the same pixels are missing in each frame). Figure 3.2 uses the same corruption scheme but we deleted 50% rather than 30% of pixel values.

These initial results are promising, though clearly there are still improvements to be made.

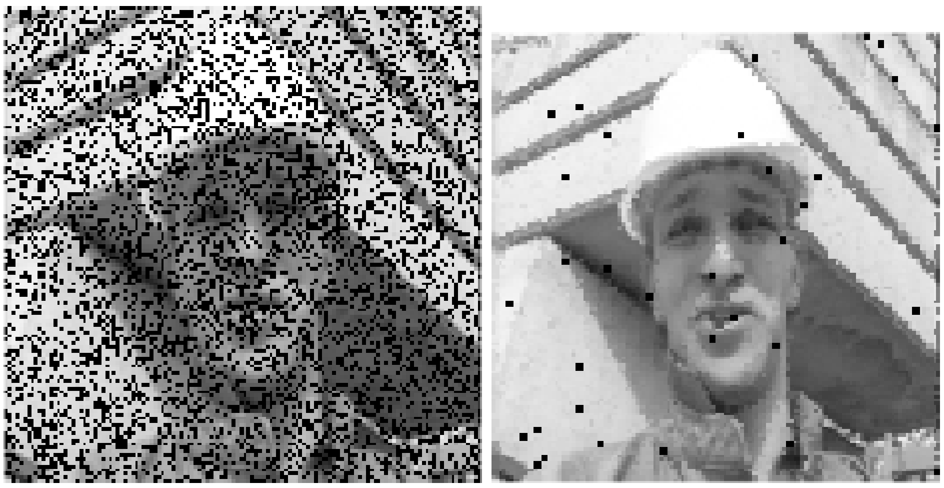


Fig. 7.1 Sample frame from corrupted video (left) and the reconstructed video (right)

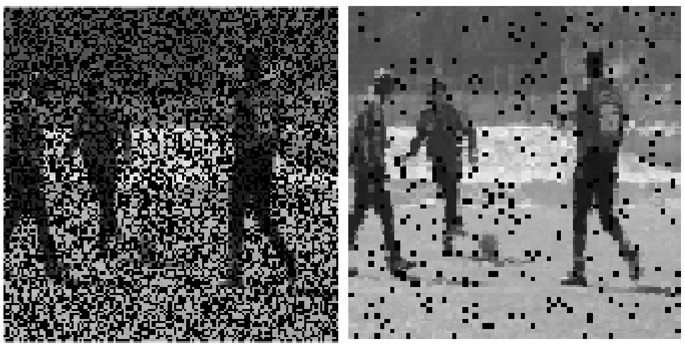


Fig. 7.2 Sample frame from corrupted video (left) and the reconstructed video (right)

Chapter 8

Conclusion

References

[1] Candès, E. J. and Wakin, M. B. (2008). An introduction to compressive sampling. *Signal Processing Magazine, IEEE*, 25(2):21–30.

[2] DeVore, R. A., Jawerth, B., and Lucier, B. J. (1992). Image compression through wavelet transform coding. *Information Theory, IEEE Transactions on*, 38(2):719–746.

[3] Pilikos, G. (2014). Signal reconstruction using compressive sensing. MPhil thesis, University of Cambridge.

[4] Shannon, C. E. (1949). Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21.

[5] Stollnitz, E. J., DeRose, T. D., and Salesin, D. H. (1995). Wavelets for computer graphics: a primer. 1. *Computer Graphics and Applications, IEEE*, 15(3):76–84.

[6] Taubman, D. and Marcellin, M. (2012). *JPEG2000 Image Compression Fundamentals, Standards and Practice: Image Compression Fundamentals, Standards and Practice*. The Springer International Series in Engineering and Computer Science. Springer US.

[7] Tipping, A. and Faul, A. (2002). Analysis of sparse bayesian learning. *Advances in neural information processing systems*, 14:383–389.

[8] Tipping, M. E. (2001). Sparse bayesian learning and the relevance vector machine. *The journal of machine learning research*, 1:211–244.

[9] Tipping, M. E., Faul, A. C., et al. (2003). Fast marginal likelihood maximisation for sparse bayesian models. In *AISTATS*.