Sparse approximation

We will consider, from several different perspectives, the problem of finding a *sparse representation* of a signal. For simplicity, the majority of our discussion will take place in the context of vectors $f \in \mathbb{R}^N$. Everything that is said here automatically extends to continuous-times signals that lie in a fixed finite-dimensional subspace in a straightforward way.

We start with a problem that has an easy solution. Say that Ψ is an orthobasis for $\mathbb{R}^N$, and we wish to find the best approximation of a a given $f \in \mathbb{R}^N$ using a fixed number of elements from Ψ. If we give ourselves a budget of S-terms, the best S-term approximation of f in the basis Ψ is defined as the solution to the following optimization program:

$$\min_{\beta \in \mathbb{R}^N} \|f - \Psi^* \beta\|_2^2 \quad \text{subject to} \quad \#\{\gamma : \beta[\gamma] \neq 0\} \leq S. \tag{1}$$

It is customary to use the notation $\|\beta\|_0$ for the number of non-zero terms in β, although it should be stressed that $\|\cdot\|_0$ does not obey the properties of a norm. Since Ψ is an orthogonal transform, Parsaval tells us that

$$\|f - \Psi^* \beta\|_2^2 = \|\Psi(f - \Psi^* \beta)\|_2^2 = \|\alpha - \beta\|_2^2$$

where $\alpha = \Psi f$ are the Ψ-transform coefficients of f. So we can rewrite the optimization program above as

$$\min_{\beta \in \mathbb{R}^N} \|\alpha - \beta\|_2^2 \quad \text{subject to} \quad \|\beta\|_0 \leq S.$$

It is easy to argue that the solution to this problem is to have β take the S-largest terms from α and set the rest to zero. Thus the algorithm for solving (1) is

1. Compute $\alpha = \Psi[f]$.

2. Find the locations of the S-largest terms in α; call this set Γ.

3. Set
$$\tilde{\beta}_S[\gamma] = \begin{cases} \alpha[\gamma] & \gamma \in \Gamma \\ 0 & \gamma \notin \Gamma \end{cases}$$

4. Compute $\tilde{f}_S = \Psi^* \tilde{\beta}_S$.

When the basis is overcomplete, choosing the best S-term approximation is nowhere near as straightforward. To separate this problem from our previous discussion (and to be consistent with the existing literature), we will change notation slightly. We have a fixed collection of functions $\phi_1, \phi_2, \ldots, \phi_p$, which we call a *dictionary*, from which we would like to choose some subset to approximate a given signal $f \in \mathbb{R}^n$. That is, we wold like to find a subset $\Gamma \subset \{1, 2, \ldots, p\}$ so that

$$f \approx \sum_{\gamma \in \Gamma} \alpha_\gamma \phi_\gamma$$

for some coefficient set $\{\alpha_\gamma,\ \gamma \in \Gamma\}$. The idea is that $p >> n$, and different signals of interest will use different subsets Γ. Given a signal, we want a systematic way of choosing a representation which uses as few terms as necessary.

Unless otherwise stated, we will use the convention that the columns of Φ are normalized, $\|\phi_k\|_2 = 1$. We will find it useful to organize the $\{\phi_k\}$ as columns in an $n \times p$ matrix Φ:

$$\Phi = \begin{bmatrix} | & | & & | \\ \phi_1 & \phi_2 & \cdots & \phi_p \\ | & | & & | \end{bmatrix}.$$

Now the task is to find a sparse vector α such that

$$f \approx \Phi \alpha.$$

We can visualize this task as selecting an appropriate subset of the columns from Φ to build up f — each subset of columns spans a different subspace of $\mathbb{R}^n$, so in some sense we choosing from a family of subspaces the one which is closest to containing f.

In general, finding the best representation in an overcomplete is a problem with combinatorial complexity (NP hard). But it can still be attacked in a principled manner. In the next two lectures, we will talk about three very different frameworks for approaching this problem:

Best basis For specialized (but very flexible) families of representations based on the LOT, Ψ has special "tree-like" structure which can be exploited to choose a best orthobasis adapted to the signal.

Greedy algorithms Matching pursuit (MP) and the closely related Orthogonal Matching Pursuit (OMP) operate by iterative choosing columns of the matrix. At each iteration, the column that reduces the approximation error the most is chosen.

Convex programming Relaxes the combinatorial problem into a closely related convex program, and minimizes a global cost function. The particular program, based on ℓ_1 minimization, we will look at has been given the name Basis Pursuit in the literature.

Best Basis

(A first look)

With the LOT, we have a very flexible framework for partitioning the real line and analyzing each segment (local trig. series in each interval)

Natural question: Given a signal $f(t)$, what is the "best" partition?

To answer this question, we will restrict ourselves to dyadic partions and a very particular meaning of "best".

Dyadic Partions

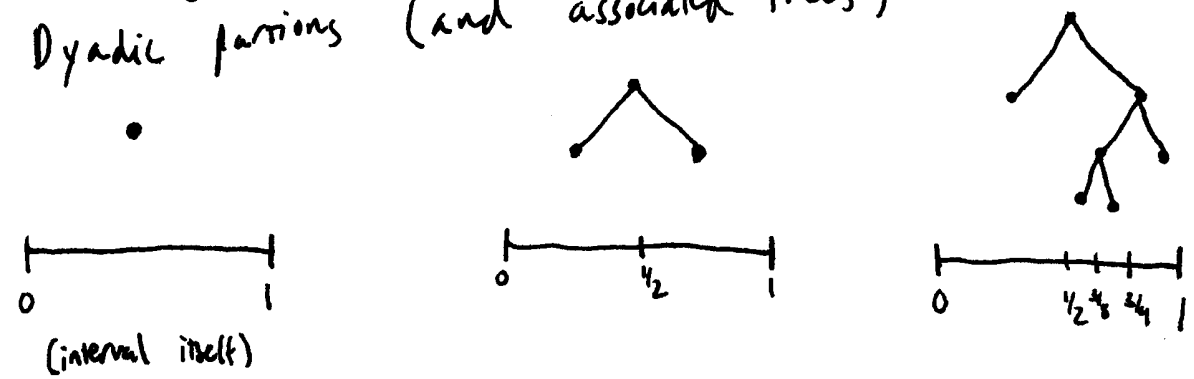
A dyadic partition of an interval is any partition we can generate by recursively splitting sub intervals in half.

Associated with each dyadic partition is a binary tree.

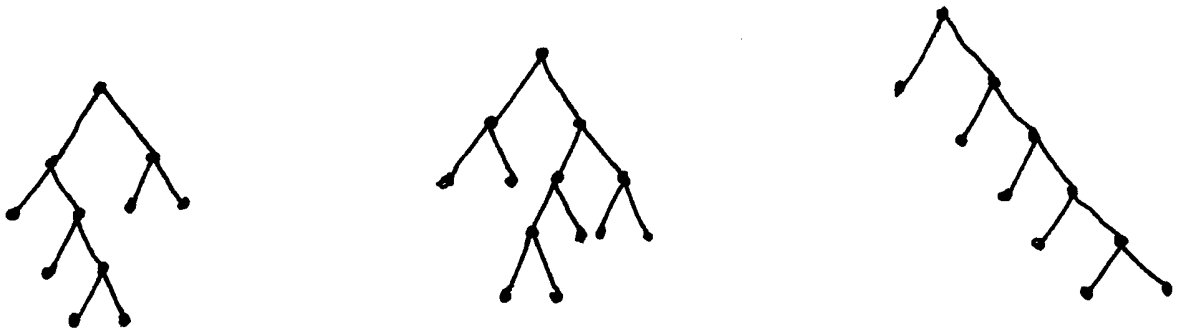
Examples.

Partitioning the interval $[0,1]$

Dyadic partions (and associated trees)



What do the partitions for these trees look like?



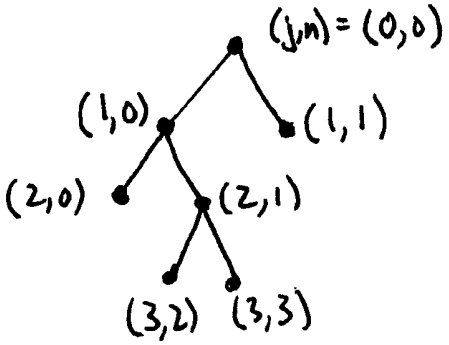
Note: To correspond to a dyadic partition each parent in the tree must have either 0 or 2 children.

Nodes with 0 children are called leaves.

Each node in a dyadic tree can be indexed by a <u>scale</u> j and a <u>shift</u> n.

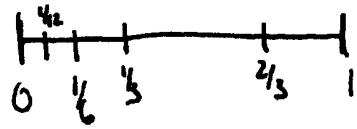
$0 \le j \le J$ $\quad J$ = maximum depth of tree

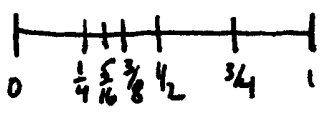
$0 \le n \le 2^j - 1$

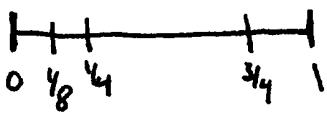


The <u>children</u> of a node indexed (j,n) are indexed by $(j+1, 2n)$ and $(j+1, 2n+1)$.

<u>Exercise</u> Which of these are <u>not</u> dyadic partions ?







Given a binary tree, the endpoints of the corresponding partition of $[0,1]$ are

$$a_{j,n} = n \cdot 2^{-j} \quad \text{for each } \underline{\text{leaf}} \ (j,n)$$

Along with a partition, each (valid) binary tree corresponds to a different orthogonal decomposition of $L_2([0,1])$.

Take $\eta \leq 2^{-J-1}$

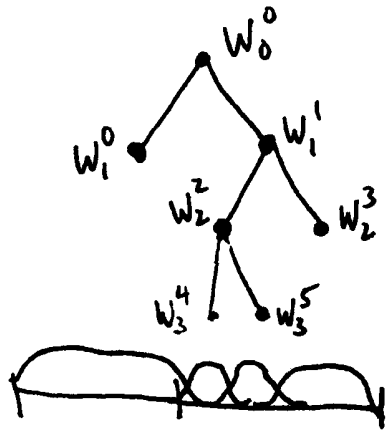
For a dyadic interval $[a_{j,n}, a_{j,n+1}] = [n \cdot 2^{-j}, (n+1) \cdot 2^{-j}]$

define the window

$$g_{j,n}(t) = \begin{cases} \beta\left(\frac{t - a_{j,n}}{\eta}\right) & t \in [a_{j,n} - \eta, a_{j,n} + \eta] \\ 1 & t \in [a_{j,n} + \eta, a_{j,n+1} - \eta] \\ \beta\left(\frac{a_{j,n+1} - t}{\eta}\right) & t \in [a_{j,n+1} - \eta, a_{j,n+1} + \eta] \\ 0 & \text{otherwise} \end{cases}$$

and the subspaces W_j as before

$f \in W_j \Leftrightarrow f(t)$ can be written $f(t) = g_{j,n}(t) \cdot h(t)$ where $h(t)$ is symmetric around $a_{j,n}$ and anti-symmetric around $a_{j,n+1}$



We can write

$$L_2([0,1]) = \bigoplus_{(j,n)\in Leaves(\Upsilon)} W_j^n$$

for any valid binary tree Υ.

Also, it is not hard to see that the children spaces W_{j+1}^{2n}, W_{j+1}^{2n+1} orthogonally decompose the parent space W_j^n. That is

$$W_j^n = W_{j+1}^{2n} \oplus W_{j+1}^{2n+1}$$

We know an orthobasis for each W_j^n, we will denote it

$$\mathcal{B}_j^n = \left\{ g_{n,j}(t) \cdot \sqrt{2^{j+1}} \cdot \cos\left[\pi(k+1/2)\cdot \frac{t - a_{n,j}}{2^{-j}}\right] \right\}_{k\in\mathbb{Z}}$$

So given a valid binary tree Υ, we have an orthobasis for $L_2([0,1])$

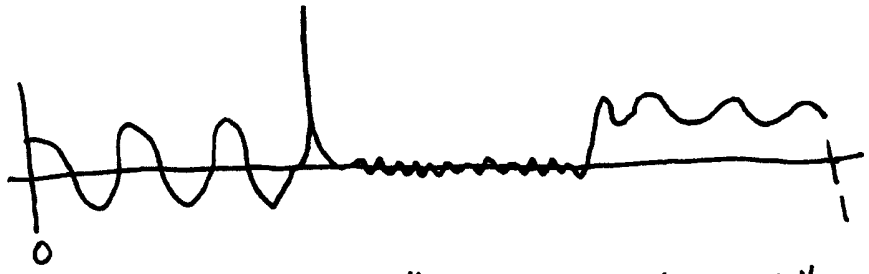
$$\mathcal{B}_\Upsilon = \bigcup_{(j,n)\in \text{Leaves}(\Upsilon)} \mathcal{B}_j^n$$

↳ "orthobasis corresponding to Υ"

Choosing a dyadic partition

How do we choose which dyadic partition is best for a given signal $f(t)$?

Suppose $f(t)$ looks like this:



What would a "good" partition look like (qualitatively)?

We want an orthobasis $\mathcal{B}_\Psi$ such that the decomposition

$$\{\langle f, \phi \rangle\}_{\phi \in \mathcal{B}_\Psi}$$

is maximally concentrated.
(i.e. it gives the simplest description of $f(t)$).

To quantify this, we will assign a cost of a basis (partition) using an entropy measure:

$$\mathcal{C}(f, \mathcal{B}_\Psi) = -\sum_{\phi \in \mathcal{B}_\Psi} \frac{|\langle f, \phi \rangle|^2}{\|f\|_2^2} \cdot \log\left(\frac{|\langle f, \phi \rangle|^2}{\|f\|_2^2}\right)$$

$\mathcal{C}(f, \mathcal{B}_\Psi)$ is minimum ($=0$) when _____

$\mathcal{C}(f, \mathcal{B}_T)$ is maximum when _____

Since the $W_j^\wedge$ are $\perp$ for $(j,n) \in \text{Leaves}(T)$ we can write

$$\mathcal{C}(f, \mathcal{B}_\Psi) = \sum_{(j,n) \in \text{Leaves}(T)} \mathcal{C}(f, \mathcal{B}_j^\wedge)$$

$$\mathcal{B}_j^\wedge = -\sum_{\phi \in \mathcal{B}_j^\wedge} \frac{|\langle f, \phi \rangle|^2}{\|f\|_2^2} \cdot \log\left(\frac{|\langle f, \phi \rangle|^2}{\|f\|_2^2}\right)$$

Associated with each node in $\mathcal{T}$ we have

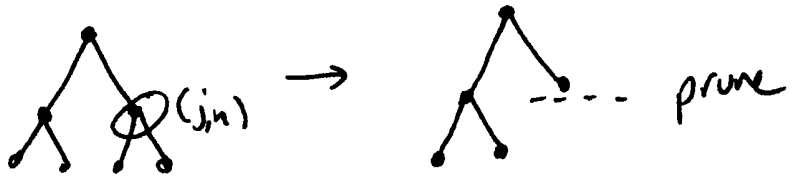
- an <u>index</u> (j,n) (scale, shift)
- a subspace W_j^n
- an orthobasis $\mathcal{B}_j^n$ (for W_j^n)
- a cost $\mathcal{C}(f, \mathcal{B}_j^n)$
  (given a signal $f(t)$)

The goal is given an $f(t)$, find the tree $\mathcal{T}$ that <u>minimizes the sum of the costs at the leaves</u>.

$$\min_{\mathcal{T}} \sum_{(j,n)\in \text{Leaves}(\mathcal{T})} \mathcal{C}(f, \mathcal{B}_j^n)$$

Of course, $\mathcal{C}(f, \mathcal{T})$ will change as we vary $\mathcal{T}$. But because the $W_j^\wedge$ are $\perp$ for $(j,n) \in \text{Leaves}(\mathcal{T})$, collapsing* a pair of children W_{j+1}^{2n} and W_{j+1}^{2n+1} into their parent $W_j^\wedge$ does not affect the contributions at the other leaves.

Collapsing or pruning a tree at node (j,n) means we use $B_j^\wedge$ to span $W_j^\wedge$ instead of $B_{j+1}^{2n} \cup B_{j+1}^{2n+1}$



This allows us to decide:
"Which is better, $B_j^\wedge$ or $B_{j+1}^{2n} \cup B_{j+1}^{2n+1}$?"
independently from decisions in other subtrees.

We can then use a fast dynamic program to find the global minimum of $\mathcal{C}(f, B_\mathcal{T})$ over all valid dyadic trees $\mathcal{T}$.

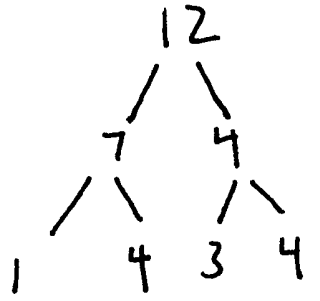
Example:

$J = 2$. Say $\mathcal{C}(f, B_0^0) = 12$

$\mathcal{C}(f, B_1^0) = 7 \quad \mathcal{C}(f, B_1^1) = 4$

$\mathcal{C}(f, B_2^0) = 1 \quad \mathcal{C}(f, B_2^1) = 4 \quad \mathcal{C}(f, B_2^2) = 3$

$\mathcal{C}(f, B_2^3) = 4$



What is the best partition?

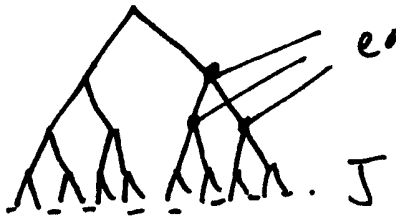
Optimization Algorithm (CART)

Maximum tree depth J

① Initialize the score S_j^n at each node on the tree with

$$S_j^n = \ell(f, \mathcal{B}_j^n)$$

Initialize the tree $\mathcal{T}$ as a full binary tree out to level J



② For $j = J-1, J-2, \ldots, 0$ and $n = 0, 1, \ldots, 2^j - 1$ do

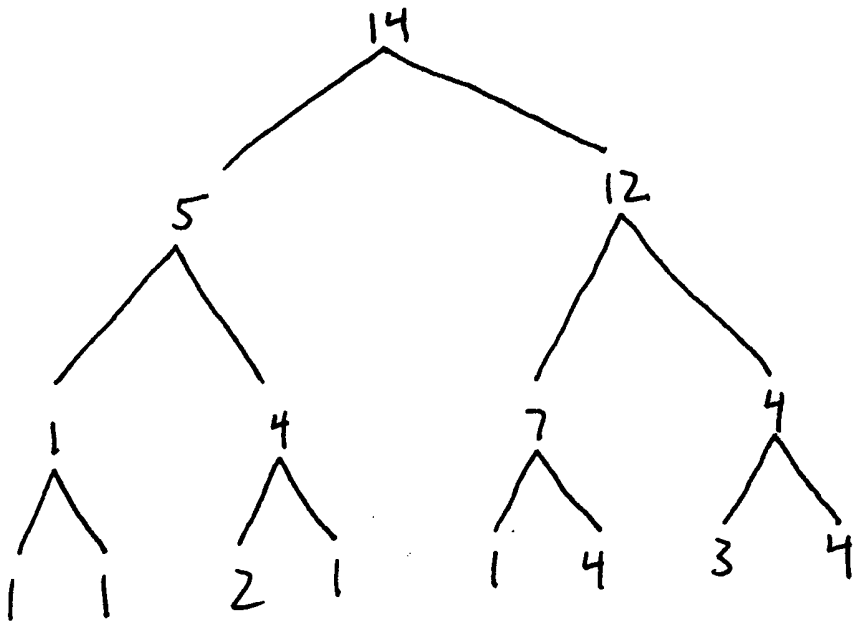
If $S_j^n \leq S_{j+1}^{2n} + S_{j+1}^{2n+1}$ then prune tree at node (j,n)

else set $S_j^n = S_{j+1}^n + S_{j+1}^{2n+1}$

After we have traveled all the way up the tree, $\mathcal{T}$ will correspond to the optimal cost/partition/basis

Example.

Suppose the $\mathcal{C}(f, B_j^n)$ are given by



Use CART to find the optimal tree / partition / basis.