An Attempt to Find an Optimal Wavelet for an Arbitrary Input Signal

Stefan Cline

San Diego State University, Department of Mathematics and Statistics

20 April 2022

Section Outline

Project Goal

I wanted to see if, given an arbitrary input signal, I could use an ML algorithm to ‘learn’ the best parameters for a predefined wavelet shape.

Project Goal

To keep things simple, I'll be using the following generic **mother wavelet** form of an equation as a sort of proof of concept

$$\hat{\psi}(t) = \exp\left(\frac{-t^2}{2}\right)(\varepsilon t + A\sin(Bt))$$

By using only odd functions being damped by a Gaussian, we satisfy (odd function times even is odd) the first condition. Dividing by the L^2 Norm we satisfy the second.

$$\underbrace{\int_{\mathbb{R}} \varphi \; dt = 0, \qquad ||\varphi||_{L^2} = 1}_{\text{Conditions}} \qquad \Rightarrow \qquad \psi(t) = \frac{\hat{\psi(t)}}{||\hat{\psi}(t)||_{L^2}}$$

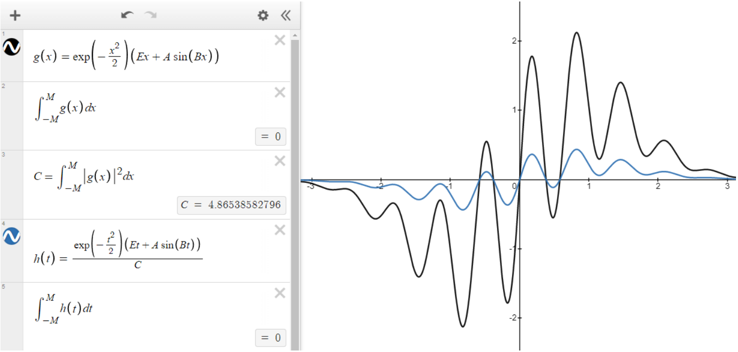
General Equation Form

Therefore, our overall analytical expression for a **daughter wavelet** indexed by $j \in (J \subset \mathbb{Z}^+)$ becomes

$$
\begin{aligned}
\varphi_j(t) &= \frac{1}{\sqrt{a^j}} \psi\left(\frac{t}{a^j}\right) \\
&= \frac{1}{\sqrt{a^j}} \frac{\exp\left(-\frac{1}{2}\frac{t^2}{a^{2j}}\right)\left(\frac{\varepsilon t}{a^j} + A\sin\left(\frac{Bt}{a^j}\right)\right)}{\left(\int_{\mathbb{R}} \exp\left(-\frac{t^2}{a^{2j}}\right)\left(\frac{\varepsilon t}{a^j} + A\sin\left(\frac{Bt}{a^j}\right)\right)^2 \, dt\right)^{1/2}}
\end{aligned}
$$

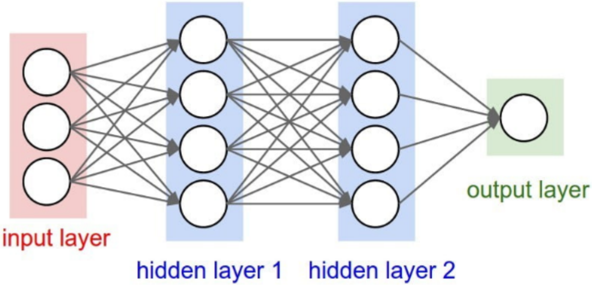
Example Mother Wavelet

For values of $\varepsilon = 1.8$, $A = 1.5$ and $B = 9.5$. The blue curve is the normalization of the black curve.



How ML Works (1)

Think of each node (circle) as containing some value and the (lines) mathematical operations performed on them produce new nodes.



How ML Works (2)

The input to the NN, $\mathbf{x}$, is fed through some function or set of functions which we'll generically call $\mathcal{N}$, produces some output $\mathbf{y}$. So,

$$\mathcal{N}(\mathbf{x}) = \mathbf{y}$$

We want to know, how well did $\mathbf{y}$ match up to our expectations?

This is determined by a **cost function**. In *supervised learning*, we already know ahead of time what we wish the answer to be. Using least squares ($D \equiv$ Desired)

$$C = (\mathcal{N}(\mathbf{x}) - \mathcal{N}_D(\mathbf{x}))^2$$

How ML Works (3)

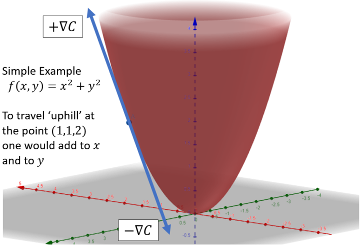
We need a way for the NN to learn. Using **gradient descent**, we determine the partial derivative per each parameter, $p_n \in \{p_1, ..., p_N\}$ to get $-\nabla C$ where

$$\nabla C = \begin{bmatrix} \frac{\partial C}{\partial p_1} \\ \vdots \\ \frac{\partial C}{\partial p_N} \end{bmatrix}$$

The big take-away here is that we are trying to **minimize our cost function**.

How ML Works (4)

Gradients give steepest ascent for a given parameter[1]. Then, simply add or subtract a small value known as the **learning rate** from the parameter to move it ‘downhill’.



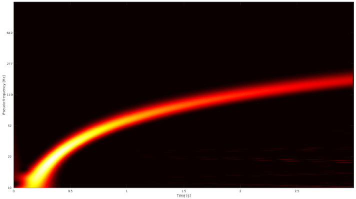
This whole process is known as **back propagation**.

[1] This is what's given to us from finding ∇C.

Challenge with Supervised Learning

What is a ‘good’ wavelet?

I noticed this classic example seemed to be zero almost everywhere except in a tight band.



I told the algorithm to learn to minimize the field, i.e., $\forall j \in (J \subset \mathbb{Z}^+), \forall t \in \Omega,\ \mathcal{N}_D(\mathbf{x}) = 0.$

Choice of $\hat{\psi}$

Recall:

$$\hat{\psi}(t) = \exp\left(\frac{-t^2}{2}\right)(\varepsilon t + A\sin(Bt))$$

This brings us back to the εt term. This will ensure that our wavelet simply doesn't learn to become zero. Cheeky little devil!

Also, we note that we'll possibly need a more robust way to ensure that our algorithm doesn't simply learn to minimize our parameters to be zero to simply be negligibly small.

Enhanced Cost Function

We'll keep the zero field idea

$$C = (\mathcal{N}(\mathbf{x}) - \mathcal{N}_D(\mathbf{x}))^2$$

but now we'll add in two biasing terms that heavily penalize the algorithm if it tries to make our parameters too small.

$$C = (\mathcal{N}(\mathbf{x}) - \mathcal{N}_D(\mathbf{x}))^2 + \mathcal{Q}_A + \mathcal{Q}_B$$

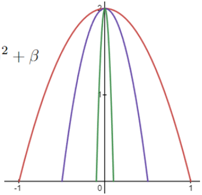
Again we note that from our form of $\hat{\psi}(t)$ our only two parameters for this example are A and B ($\Gamma \in \{A, B\}$).

$\mathcal{Q}_\Gamma = -(l^{-1}\sqrt{\beta}\Gamma)^2 + \beta$

$l \equiv$ limit.

$\beta \equiv$ added error.

$\Gamma \equiv$ parameter.



Forward Propagation (1)

We know that in the forward direction we'll generate a **scalogram** by computing

$$\forall j \in J, \qquad \varphi_j(t) * f(t) = \int_{\mathbb{R}} \varphi_j(t-\tau) * f(\tau)\ d\tau$$

Therefore, on this heinous equation, I needed to perform back propagation. Many details later...

Forward Propagation (2)

Note here we still need to do $\int_{\mathbb{R}} W \; d\tau$ as this is the convolution[2]

$$W := \frac{e^{-\frac{(t-\tau)^2}{2 a^{2j}}} \left(\frac{\varepsilon (t-\tau)}{a^j} + A \sin\left(\frac{B (t-\tau)}{a^j} \right) \right) f(\tau)}{\sqrt{a^j} \sqrt{\int_{-\infty}^{\infty} e^{-\frac{(t-\tau)^2}{a^{2j}}} \left(\frac{\varepsilon (t-\tau)}{a^j} + A \sin\left(\frac{B (t-\tau)}{a^j} \right) \right)^2 d\tau}}$$

[2]The outside integral is irrelevant to the calculations but was too computationally excessive for Maple so I left it out.

Forward Propagation (3)

Each discrete point in the field/scalogram is a convolution

$$
\begin{array}{c c}
j_0 & \multirow{4}{*}{$\left[\begin{array}{ccc}
W_{t_0,j_0} = \int_\Omega \varphi(t_0-\tau)f(\tau)d\tau & \cdots & W_{t_0,j_0} = \int_\Omega \varphi(t_T-\tau)f(\tau)d\tau \\
W_{t_0,j_1} = \int_\Omega \varphi(t_0-\tau)f(\tau)d\tau & & \\
\vdots & \ddots & \\
W_{t_0,j_J} = \int_\Omega \varphi(t_0-\tau)f(\tau)d\tau & & W_{t_T,j_J} = \int_\Omega \varphi(t_T-\tau)f(\tau)d\tau
\end{array}\right]$} \\
\Big\downarrow & \\
& \\
J & \\
& t_0 \xrightarrow{\hspace{10em}} T
\end{array}
$$

Each point was calculated because nicely we know
$\hat{h}[k] = \hat{f}[k]\hat{g}[k] = \widehat{f \otimes g}[k]$

Analytical Back Propagation - Finding ∇C (1)

Again here we still need to do $\int_{\mathbb{R}} EQA \; d\tau$

> $EQA := diff(W, A)$

$$EQA := \frac{e^{-\frac{(t-\tau)^2}{2a^{2j}}} \sin\left(\frac{B(t-\tau)}{a^j}\right) f(\tau)}{\sqrt{a^j}\sqrt{\int_{-\infty}^{\infty} e^{-\frac{(t-\tau)^2}{a^{2j}}}\left(\frac{\varepsilon(t-\tau)}{a^j} + A\sin\left(\frac{B(t-\tau)}{a^j}\right)\right)^2 d\tau}} - \left(e^{-\frac{(t-\tau)^2}{2a^{2j}}}\left(\frac{\varepsilon(t-\tau)}{a^j} + A\sin\left(\frac{B(t-\tau)}{a^j}\right)\right) f(\tau)\left(\int_{-\infty}^{\infty} 2\, e^{-\frac{(t-\tau)^2}{a^{2j}}}\left(\frac{\varepsilon(t-\tau)}{a^j} + A\sin\left(\frac{B(t-\tau)}{a^j}\right)\right)\sin\left(\frac{B(t-\tau)}{a^j}\right) d\tau\right)\right) \Bigg/ \left(2\sqrt{a^j}\left(\int_{-\infty}^{\infty} e^{-\frac{(t-\tau)^2}{a^{2j}}}\left(\frac{\varepsilon(t-\tau)}{a^j} + A\sin\left(\frac{B(t-\tau)}{a^j}\right)\right)^2 d\tau\right)^{3/2}\right)$$

Analytical Back Propagation - Finding ∇C (2)

Again we still need to do $\int_{\mathbb{R}} EQB \; d\tau$

> $EQB := diff(W, B)$

$$EQB := \frac{e^{-\frac{(t-\tau)^2}{2 a^{2j}}} A\,(t-\tau) \cos\left(\frac{B\,(t-\tau)}{a^j}\right) f(\tau)}{\left(a^j\right)^{3/2} \sqrt{\int_{-\infty}^{\infty} e^{-\frac{(t-\tau)^2}{a^{2j}}} \left(\frac{\varepsilon\,(t-\tau)}{a^j} + A \sin\left(\frac{B\,(t-\tau)}{a^j}\right)\right)^2 d\tau}} - \left(e^{-\frac{(t-\tau)^2}{2 a^{2j}}} \left(\frac{\varepsilon\,(t-\tau)}{a^j} + A \sin\left(\frac{B\,(t-\tau)}{a^j}\right)\right) f(\tau) \left(\int_{-\infty}^{\infty} \frac{2\, e^{-\frac{(t-\tau)^2}{a^{2j}}} \left(\frac{\varepsilon\,(t-\tau)}{a^j} + A \sin\left(\frac{B\,(t-\tau)}{a^j}\right)\right) A\,(t-\tau) \cos\left(\frac{B\,(t-\tau)}{a^j}\right)}{a^j} d\tau \right) \right) \Bigg/ \left(2 \sqrt{a^j} \left(\int_{-\infty}^{\infty} e^{-\frac{(t-\tau)^2}{a^{2j}}} \left(\frac{\varepsilon\,(t-\tau)}{a^j} + A \sin\left(\frac{B\,(t-\tau)}{a^j}\right)\right)^2 d\tau \right)^{3/2} \right)$$

Analytical Back Propagation - Finding ∇C (3)

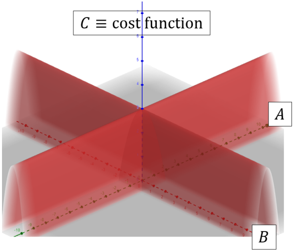
```python
# num = numerator, dem = denominator
def EQA(neg_oh,t,tau,two,thd2,a_to_j,A,B,f_t,delta_t,eps):
  Q1 = (t-tau)/a_to_j
  a_to_j2 = a_to_j**two
  S1 = A*np.sin(B*Q1)
  # usin em
  num1 = np.exp(neg_oh*(t-tau)**two/(a_to_j2))*np.sin(B*Q1)*f_t
  dem1_intg = np.exp(-(t-tau)**two/a_to_j2)*(eps*Q1+S1)**two
  dem1_integral = integrate.simpson(dem1_intg,dx=delta_t)
  dem1 = np.sqrt(a_to_j)*np.sqrt(dem1_integral)
  T1 = num1/dem1
  LT = np.exp(neg_oh*(t-tau)**two/(two*a_to_j2))*(eps*Q1+S1)*f_t
  RT_intg = two*np.exp(-(t-tau)**two/a_to_j2)*(eps*Q1+S1)*np.sin(B*Q1)
  RT = integrate.simpson(RT_intg,dx=delta_t)
  dem2_intg = np.exp(-(t-tau)**two/a_to_j2)*(eps*Q1+S1)**two
  dem2 = two*np.sqrt(a_to_j)*(integrate.simpson(dem2_intg,dx=delta_t))**(thd2)
  A_curve = T1-LT*RT/dem2
  return integrate.simpson(A_curve,dx=delta_t)


def EQB(neg_oh,t,tau,two,thd2,a_to_j,A,B,f_t,delta_t,eps):
  Q1 = B*(t-tau)/a_to_j
  Q2 = eps*(t-tau)/a_to_j
  Trigs = A*np.sin(Q1)
  Trigc = A*(t-tau)*np.cos(Q1)
  # now to use em
  num1 = np.exp(neg_oh*(t-tau)**two)*Trigc*f_t
  dem1_intg = np.exp(-(t-tau)**two/a_to_j**two)*(Q2+Trigs)**two
  dem1 = a_to_j**thd2*np.sqrt(integrate.simpson(dem1_intg,dx=delta_t))
  LT = np.exp(neg_oh*(t-tau)**two/a_to_j**two)*(Q2+Trigs)*f_t
  RT_intg = two*np.exp(-(t-tau)/a_to_j)*(Q2+Trigs*Trigc)
  RT = integrate.simpson(RT_intg,dx=delta_t)
  dem2_intg = np.exp(-(t-tau)**two/a_to_j**two)*(Q2+Trigs)**two
  dem2 = two*np.sqrt(a_to_j)*(integrate.simpson(dem2_intg,dx=delta_t))**thd2
  B_curve = num1/dem1-LT*RT/dem2
  return integrate.simpson(B_curve,dx=delta_t)
```

Gross $\rightarrow$

Quick Note on Parameter Space

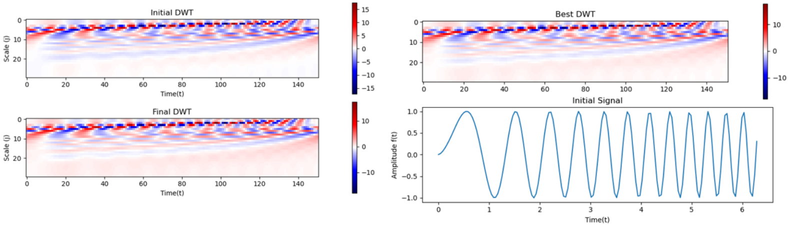
Because we did a lot of work to make sure we wouldn't simply learn $A = B = 0$, we effectively took our parameter space and did the following.

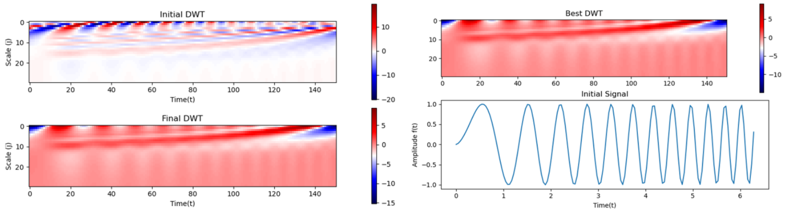


Therefore, while testing, I had to be cognizant of which of the 4 sections I was starting in $(A^+B^+,\ A^-B^+,\ A^+B^-,\ A^-B^-)$

Mixed Results for Chirp (1): $f(t) = \sin\left(4t^{1.6}\right)$

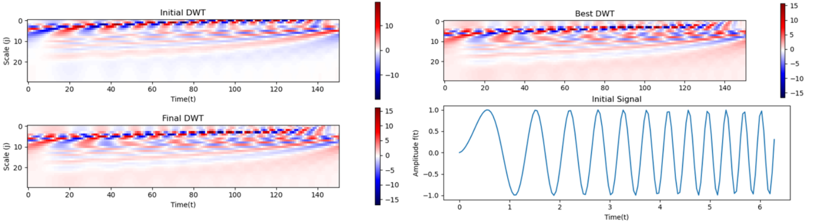
Top: A^+, B^+, Bottom: A^+, B^-

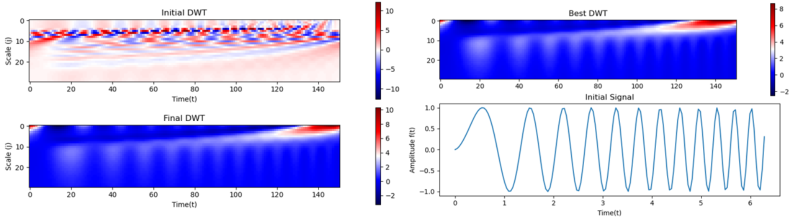




Mixed Results for Chirp (2): $f(t) = \sin\left(4t^{1.6}\right)$

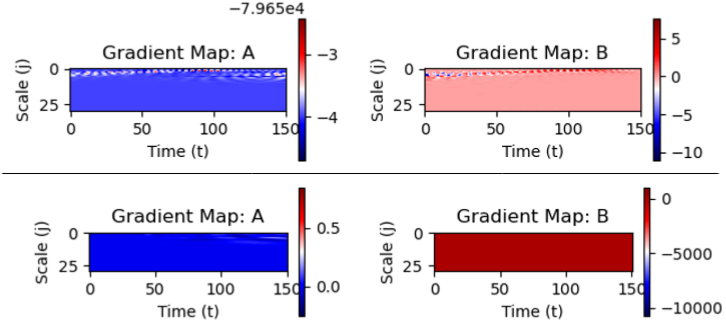
Top: A^-, B^+, Bottom: A^-, B^-





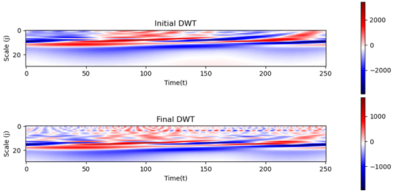
Verification of Minimization

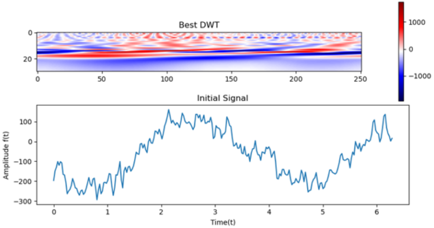
Did the algorithm actually do what I wanted it to?
Top: A^+, B^+, Bottom: A^+, B^-



Arbitrary Signal - Seismic Data

I took this signal from Dr. Gilles' seismic data (available in MatLab). From an eyeball perspective, this did seem to work decently well, and gave the DWT a more structured appearance while also minimizing the field.

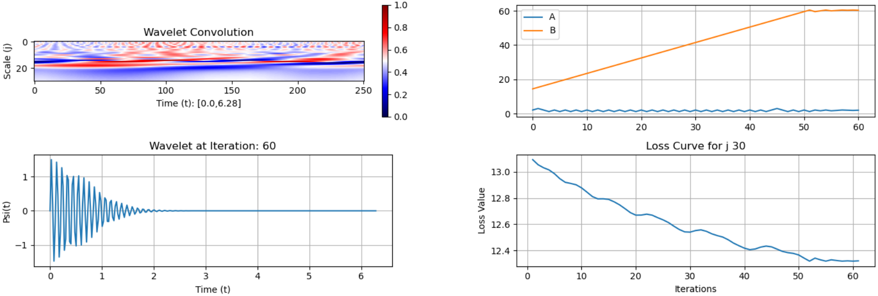




Optimized to a Non-Trivial Curve

Here we see all of this nicely come together.

1. The loss curve goes down.
2. Clear difference in wavelet shape from $A_0, B_0 \rightarrow A_{63}, B_{63}$
3. The scalogram looks more organized



Concluding Thoughts

o This method was finicky and required several iterations. Manual verification was often required. However, some results did seem promising/interesting.

o I would be curious to know what an expert in this space would have to say about my final results.

o More complex potential wavelet forms both real and complex possible (where $\mathcal{O} \equiv$ odd functions)

$$\hat{\psi}(t) = \exp\left(\frac{-t^2}{2}\right)\left(\sum_{n=1}^{N} A_n \sin\left(B_n t\right) + \sum_{n=1}^{N} E_n t^{2n-1} + \sum \mathcal{O}\right)$$

Questions?

References

1. Jerome Gilles (2023). Empirical Wavelet Transforms (https://www.mathworks.com/matlabcentral/fileexchange/42141-empirical-wavelet-transforms), MATLAB Central File Exchange. Retrieved April 25, 2023.
2. Jerome Gilles (2023). Course Reader for Math 668 - Fourier Analysis. San Diego State University.

Links

PUT LINKS TO YOUR CODE HERE