Time variable prediction without mathematics

Jon Claerbout

ABSTRACT

With almost no math, a quick trick leads to ℓ_1 norm nonstationary decon.

SUMMARY AND CONCLUSION

Start with data (a signal of thousands of values). Start with any filter $\mathbf{f}$ of maybe ten lags. The filter will change as we move it along the data. At time t set the filter down on the data. The ten data values under the filter are designated $\mathbf{d}$. Take ϵ to be a tiny scalar (for example $\epsilon = 1/(200\,\|\mathbf{d}\|)$).

The filter $\mathbf{f}$ has output $(\mathbf{f} \cdot \mathbf{d})$ that we may choose to be a prediction of anything, for example, a prediction of the next data value to slide under the range of $\mathbf{d}$. The augmented filter $\mathbf{f} \pm \epsilon\,\mathbf{d}$ offers us the two predictions $(\mathbf{f} \pm \epsilon\,\mathbf{d}) \cdot \mathbf{d} = (\mathbf{f} \cdot \mathbf{d}) \pm \epsilon\,(\mathbf{d} \cdot \mathbf{d})$. Comparing these predictions to the actual incoming data value reveals which sign for ϵ better improves the prediction. Update the filter $\mathbf{f}$. Move to time $t + \Delta t$. Update $\mathbf{d}$. Repeat indefinitely. The filter adapts to best make your prediction.

THEORY AND POTENTIAL APPLICATIONS

The above idea can be based on conventional mathematics. The gradient of ℓ_2 normed prediction error turns out to be $\mathbf{d} \times \mathrm{PredictionError}(\mathbf{d})$. The above algorithm marches with $\mathbf{d} \times \mathrm{Signum}(\mathrm{PredictionError}(\mathbf{d}))$ which smells like ℓ_1 norm decon. Wow!

Taking many small steps down a gradient has two advantages over analytic solutions: (1) it allows nonstationarity, and (2) it streams data (saving memory).

As with deconvolution on a helix, this method extends naturally to higher dimensional spaces (such as (t, x)-space).

I'm always trying to convince students to use PEFs to make their residuals IID.

I've explained how this method extends to multichannel data (vector-valued data), but it's not yet been tried. `http://sep.stanford.edu/sep/jon/VectorDecon.pdf`