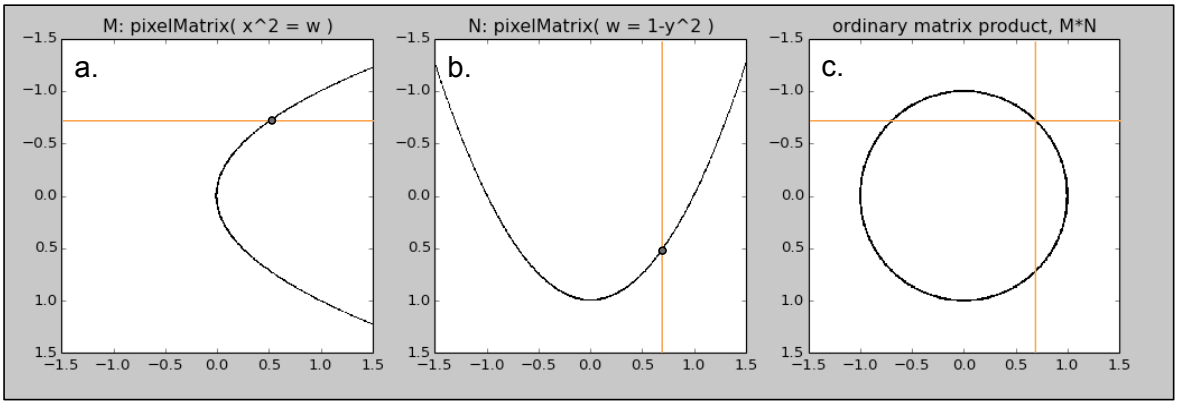
Pixel matrices for big, messy, real-world data

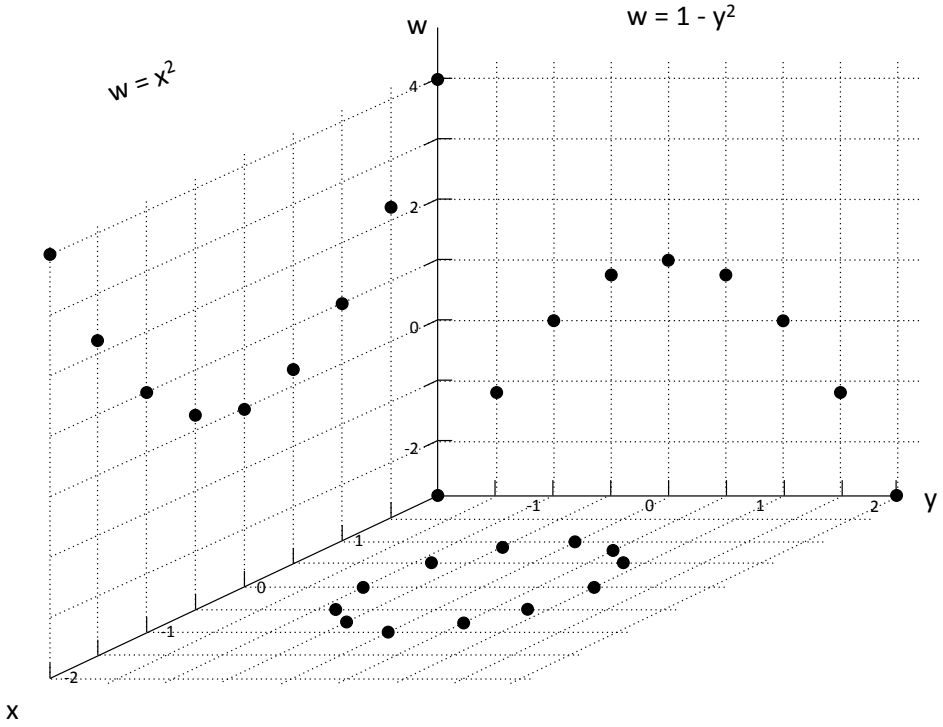
David I. Spivak

5th Mini-Symposium on Computational Topology
2016/06/15

Abstract

Natural, real-world data does not come equipped with mathematical equations that it satisfies. A data set is just a collection of observed relationships, so it is naturally messy and probabilistic. When combining and analyzing these observed relationships, it is destructive, yet common practice, to first "normalize" the data by choosing models, fitting curves, removing outliers, or estimating parameters. It would often be preferable to forgo the clean-up step and simply combine the data sets directly. Pixel matrices are an applied category-theoretic technique for doing just that. We will explain how pixel matrices offer a massively parallelizable method for approximating the solution set for systems of nonlinear equations, and how the same idea can be applied when working with messy data.


M: pixelMatrix(x^2 = w)
N: pixelMatrix(w = 1-y^2)
ordinary matrix product, M*N
a.
b.
c.


w
w = 1 - y2
w = x2
y
x

Simple version

I. Matrix multiplication $x^2 = w$ and $w = 1 - y^2$: solves the system.

II. Equivalence between Rel and Mat(Bool)

   A. Characteristic function

   B. Composition and identities

   C. Monoidal structure

   D. Intro to wiring diagrams: orientations and shadowbox

III. Pixelation (use equipment notation)

   A. $\mathbb{R} \to \underline{n}$ and refinement

   B. Closed subsets are approximated.

IV. Handling arity: operad of cospans

   A. Solving bigger systems

   B. Wiring diagrams and variable sharing

V. Changing the semiring; messy data

VI. Computational complexity and clustering

   A. Complexity: exponential in number of links

   B. Cost is parallelizable: compositionality and clustering

Older, fleshed-out version; different order

I. Motivating example

A. Draw $x^2 = w$ and $w = 1 - y^2$, and their matrix product

B. Explain why it works (assuming continuum matrices)

1. High-level: Relations are equivalent to matrices with Boolean semiring.
2. Concrete: AND's and OR's

C. Draw the two planes and the shadow idea.

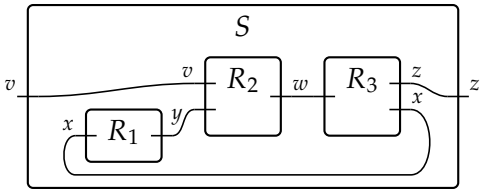
D. Outline of talk (see below)

II. Wiring diagrams

A. More generally, we have a system of equations

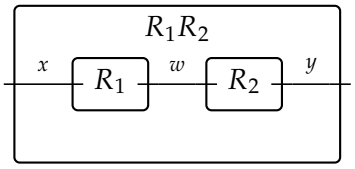
$$\begin{aligned} R_1 : &\quad x^2 + 3|x - y| - 5 = 0 \\ R_2 : &\quad y^2 v^3 - w^5 \le 0 \\ R_3 : &\quad \exists b.\ \cos(b + zx) - w^2 = 0 \end{aligned} \tag{1}$$

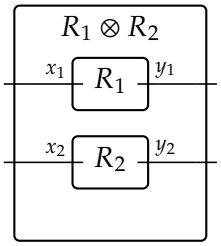
B. Draw each as a box, connect via shared variables

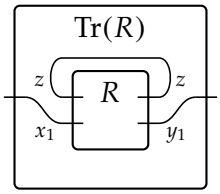


(2)

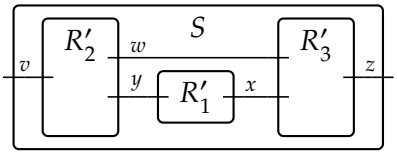
C. Formulas: break down into series, parallel, feedback







1. $S = \mathrm{Tr}\big((I_x \otimes R_1 \otimes I_v)(I_x \otimes R_2)(I_x \otimes R_3)\big)$
2. Orientations are meaningless and get in the way.



3. $S = R_2'(R_1' \otimes I_w)R_3'$
4. There is an array formula that simultaneously generalize all of these.
5. An unoriented wiring diagram is a cospan $\bigsqcup_i X_i \to V \leftarrow Y$

III. Pixelation

A. The drawings on your computer screen are matrices.

$$\begin{matrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{matrix}$$

B. Approximating a variable corresponds to a function $[a_0, a_1] \to \{1, 2, \ldots, n\} = \underline{n}$.

C. Implied pixelations: $R \subseteq A \times B$ implies $R' \subseteq \underline{a} \times \underline{b}$

1. True negatives
2. "Cocartesian": minimal implied pixelation.
3. Under wiring diagrams, cocartesianness does not compose, but implication does.

D. Clustering and computational complexity

1. The more clustering the better.

IV. Real data

A. Changing the semiring

1. Using $\mathbb{N}$ rather than Bool: number, rather than existence, of solutions
2. Using $\mathbb{R}_{\geq 0}$: densities

B. Data with error bars

C. No need for curve fitting