University of St Andrews | FOUNDED 1413

When things get noisy

Programming in the presence of ubiquitous uncertainty

Lei Fang and Simon Dobson
School of Computer Science, University of St Andrews UK

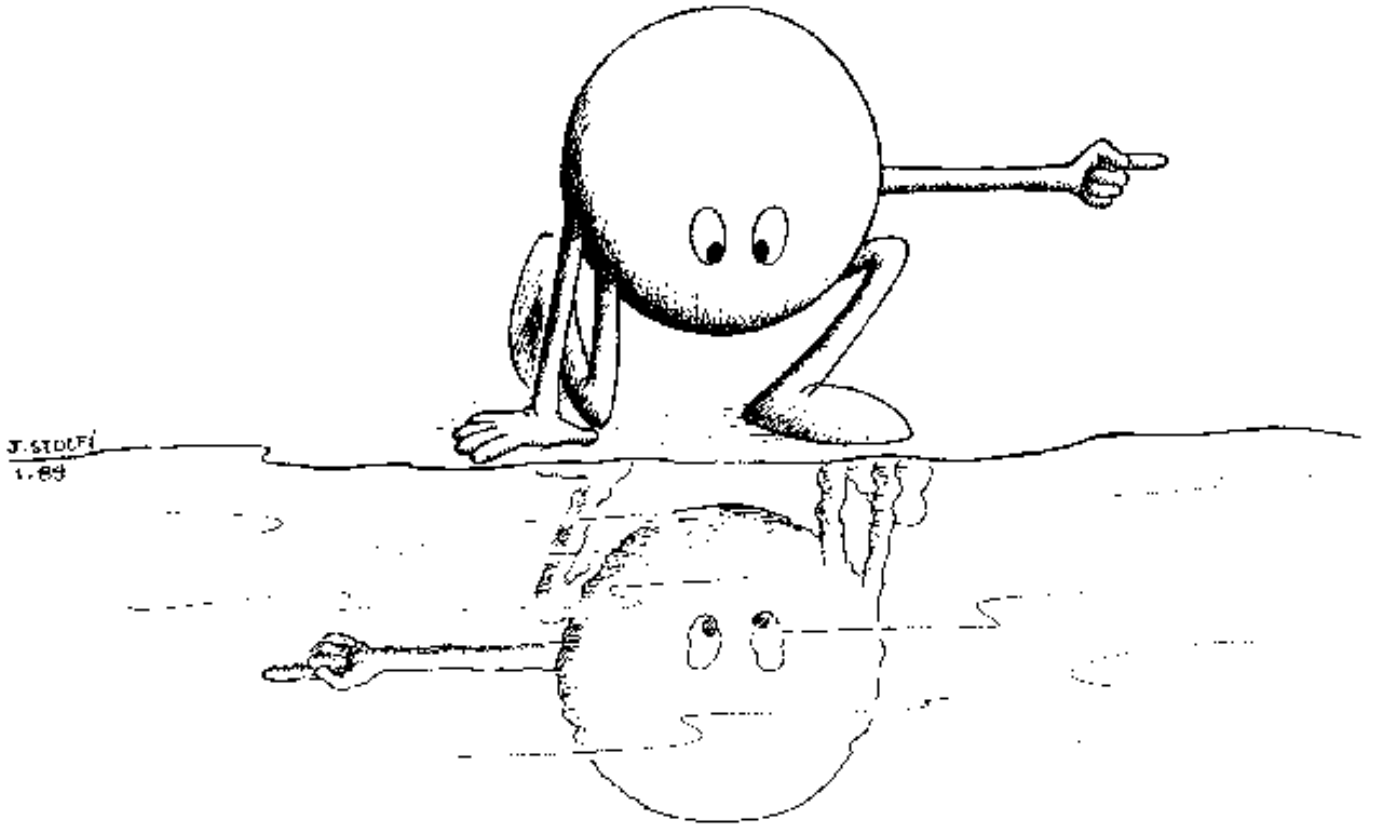
simon.dobson@st-andrews.ac.uk
http://www.simondobson.org

Introduction

- Sensor-driven systems
  - The main (only?) input stream is garbled
  - The authors of one famous experiment (Great Duck Island, 2002) deemed 30—60% of sensor data faulty
- How do you program with this level of junk?

- Re-conceptualise sensors as evidence-providers rather than data- or value-providers
  - Use to confirm/refute model hypotheses
  - Learn the distributions being observed

Sensing should be easy, but...



In theory, there is no difference between theory and practice. But, in practice, there is.

Jan L.A. van de Snepscheut

Why are sensors so bad?

- An inherent uncertainty that can't be engineered out of a system
  - Physical degradation
  - Occlusion and fouling
  - Positional uncertainty
  - Interference, accidental or deliberate

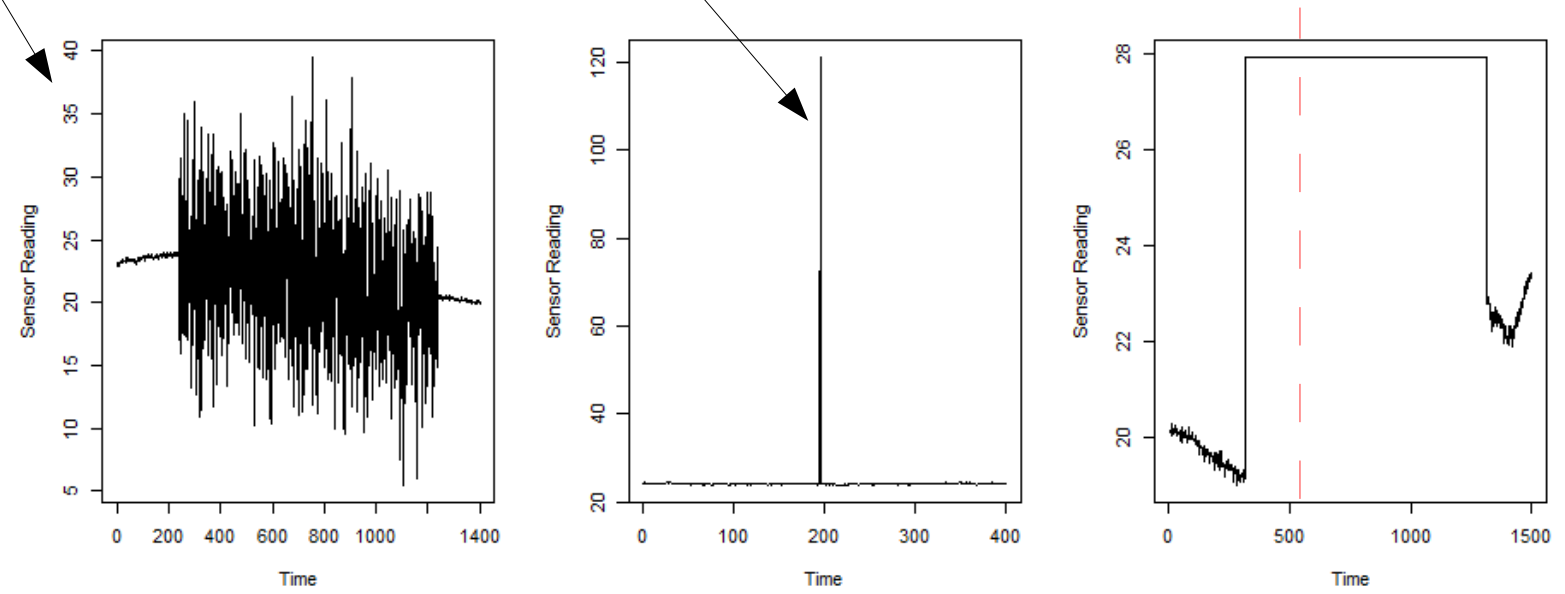
- *Physical* issues that give rise to *faults* in the data
  - Change over time, need autonomic management

Fault types

Natural variation plus noise

Not likely in your data?

Can we tell here that this is an extended fault? Or is it a change of phenomenon?



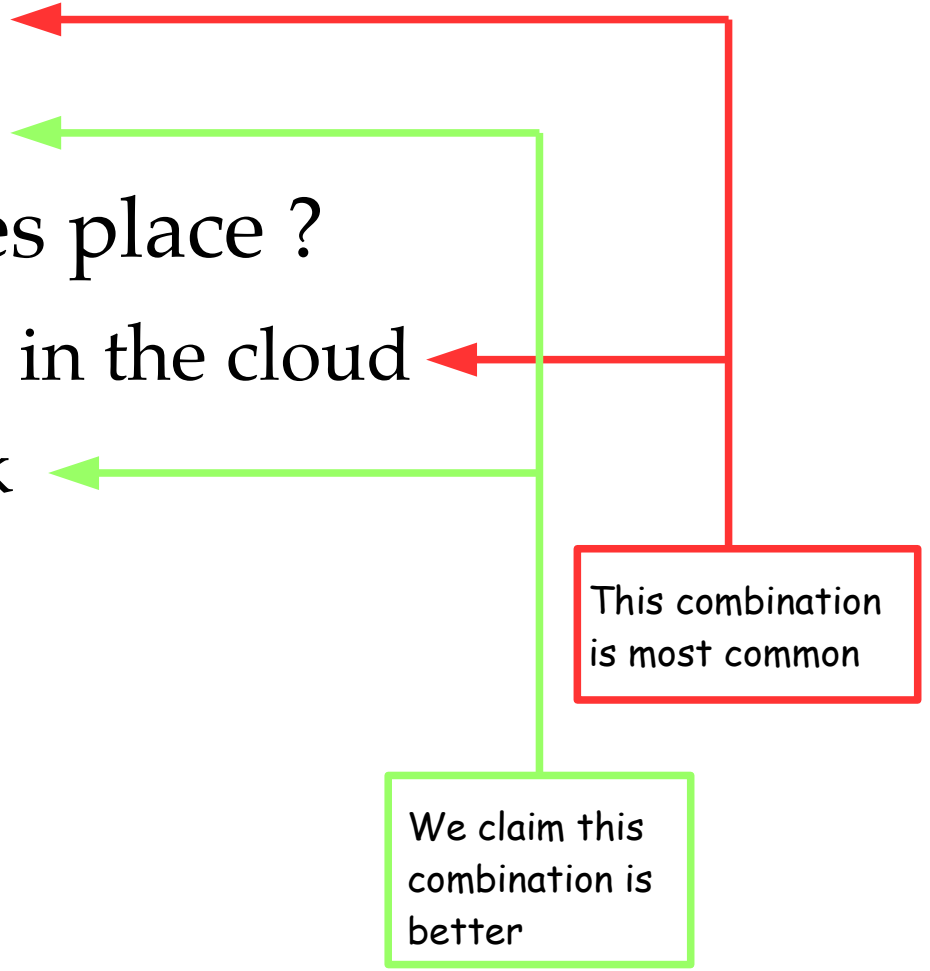
- Noise in the environment and the electronics
- Point (or wider) spikes
- De-calibration (drift) in space and time

The taxonomy of fault detectors

- When the detection takes place ?
  - Off-line, during analysis
  - On-line, during capture
- Where the detection takes place ?
  - Server-side, at the sink or in the cloud
  - Front-end, in the network

The taxonomy of fault detectors

- When the detection takes place ?
  - Off-line, during analysis
  - On-line, during capture
- Where the detection takes place ?
  - Server-side, at the sink or in the cloud
  - Front-end, in the network



Why?

- Scalable
  - Use the resources you have, more or less
- Flexible
  - Not every application sends data to the sink
  - Those that do can reduce overheads
- Robust
  - Alerts as they happen
- Tolerant
  - Dataset useful even if some nodes fail

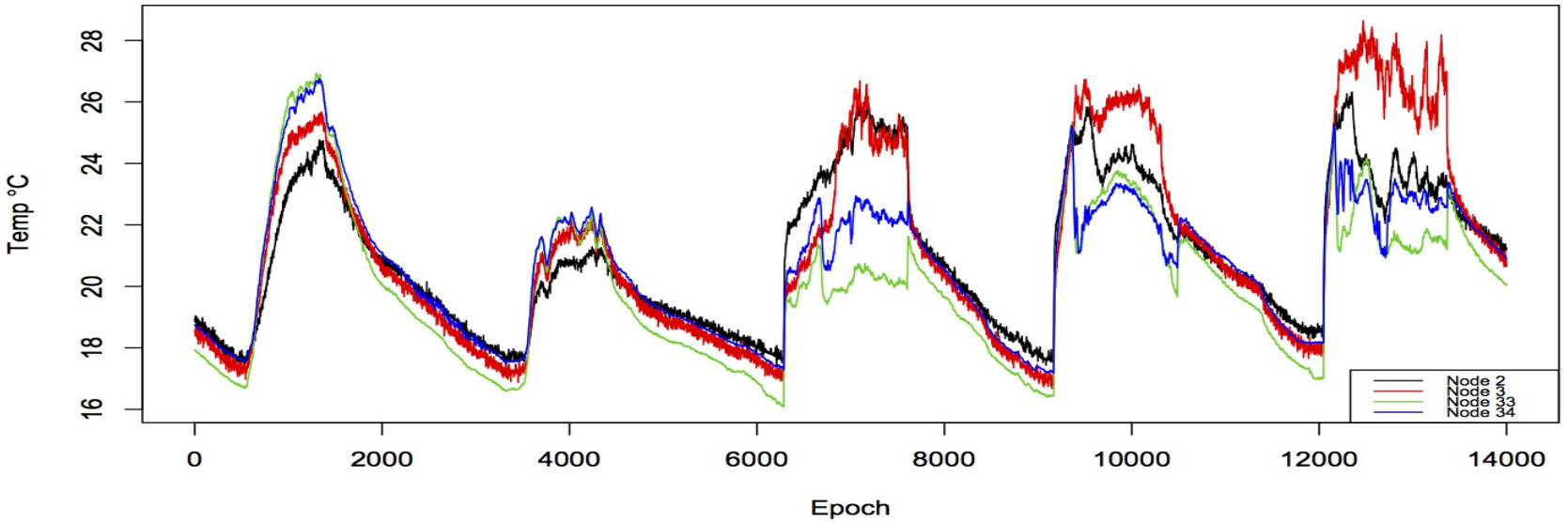
Scientific analysis based around a specific set of nodes is fragile in the face of node failure

Approach

- Hypothesise the intended data
  - *Non-stationary:* summary statistics change over time
- Sensor readings *confirm* or *refute* the hypothesis
  - Add *evidence* rather than providing *data*
  - Goal of the network is to *adapt the model* to reflect the conditions being observed on an on-going basis
  - Result is a *distribution* learned from the data
- Use model correlations to let nodes *verify* each others' readings

Spatial correlation – 1

- Neighbouring nodes observe the same trend



- Look at the differences between them to learn the ways in which the true signal is being convolved with noise

Spatial correlation – 2

- A spatial model
  - The synchronised differences are Gaussians

$$e_t \overset{iid}{\sim} \mathcal{N}(\delta_{ij}, \sigma_e^2) \qquad \{e_t = Y_{i,t} - Y_{j,t}, t \geq 0\}$$

  - Sometimes these can be derived from context
    - Sensor measurement variance may be known

$$\sigma_e^2 = \mathrm{Var}[e_t] = \mathrm{Var}[Y_i] + \mathrm{Var}[Y_j] = \sigma_i^2 + \sigma_j^2$$

Known hardware measurement error

    - In general unknown, but can be learned

Two inference problems

- Verifier node selection
  - Whether a spatial relationship exists
  - Solved by inferring the *posterior*

Geographical proximity does not necessarily lead to spatial correlation, *e.g.*, two nodes in two rooms

Mean of the difference

$$\delta_{ij} | \mathcal{D}_N$$

Sliding window history of N observations

- Fault detection works by predicting the error term and seeing whether observation agrees
  - Infer the *predictive*

$$e_{n+1} | \mathcal{D}_N$$

So we're using *observations* to learn the (posterior probabilities of) the *model* of what the observations *should* look like

Bayesian Sequential Learning

- Learning = sequential model update

$$p(\theta|\mathcal{D}_n) = p(\theta|\mathcal{D}_{n-1}, e_n)$$
$$\propto p(e_n|\mathcal{D}_{n-1}, \theta)p(\theta|\mathcal{D}_{n-1}) \quad \text{By Bayes' theorem}$$
$$= \underbrace{p(e_n|\theta)}_{\text{likelihood}} \underbrace{p(\theta|\mathcal{D}_{n-1})}_{\text{prior}} \quad \text{By conditional independence derived from the model}$$

The model given what's been observed up to (and including) now

Update

The model given what's gone before (existing model)

Start from a *sound prior* with no observations, just reflecting the model's view of what will happen

- Two cases depending on whether σ_e^2 is known
  - Use learning to narrow the width of the distribution

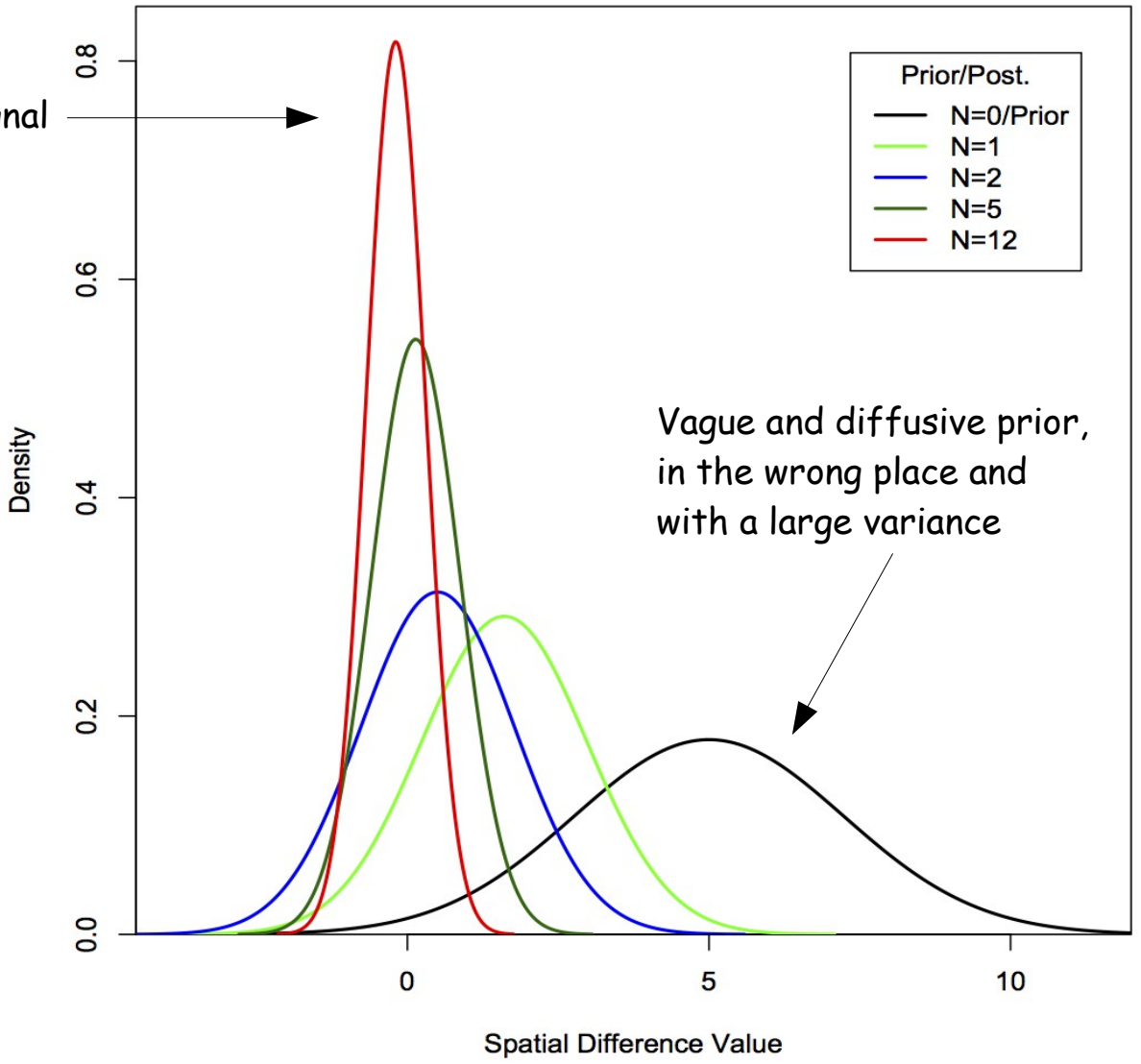
Why Bayesian learning?

- Formal statistical learning
  - Provides sound inference
- Efficient and lightweight
  - Constant space complexity
  - Constant complexity update
- Robust
  - Test incoming data against the predicted distribution
  - Only then *admit* data for future learning

No need to store any learning data over a protracted period

Learning as sequential update

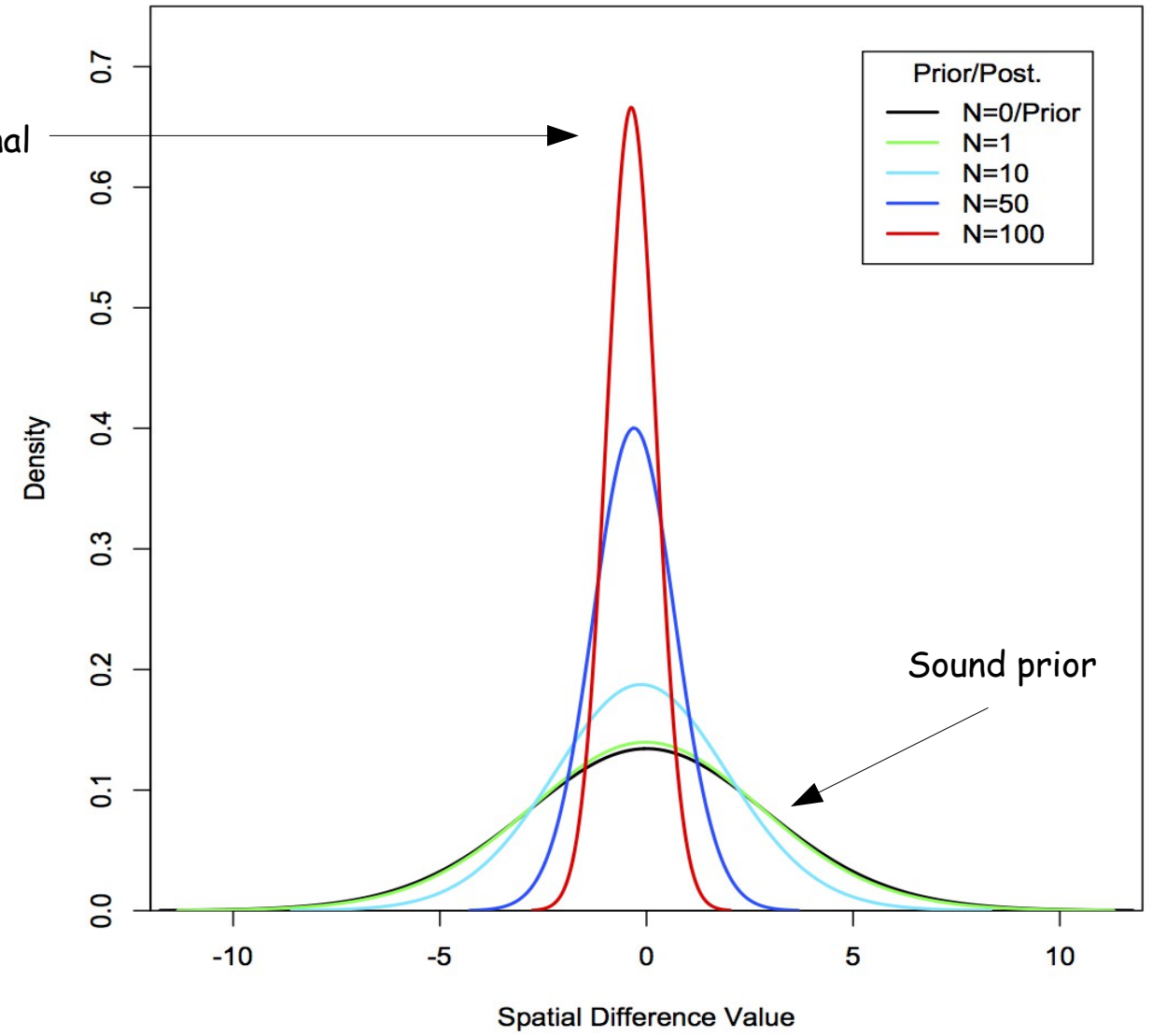
Converging to the true signal



Fang and Dobson. Fault detection based on hierarchical physical models. ACM TOSN. In preparation

Learning the variance

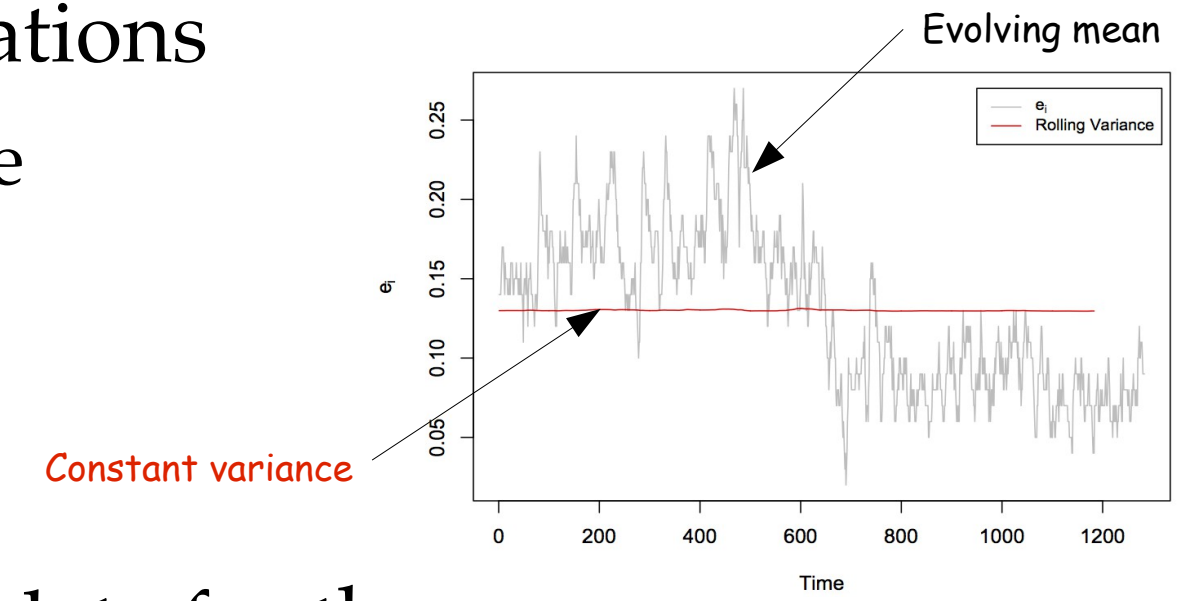
Converging to the true signal



The curves are Student T distributions, with more space under the tails for outliers

Model update

- Track the observations
  - Spatial difference is evolving



- Time-varying update for the mean
  - Recent data is given higher weight

$$\tilde{m}_n = \tilde{m}_{n-1} + \psi(e_n - \tilde{m}_{n-1})$$

Exponential smoothing, the sum of the weights is 1, leading to an unbiased estimator

Smoothing parameter between 0 and 1

Protocol design

- Group spatial verification
  - Don't return data consistent with the model
  - It's adding no information
- Resilient to broken-down spatial correlation
  - Pairwise correlations tend to change

- ...at a reasonable overhead in term of the extra comms and computation required

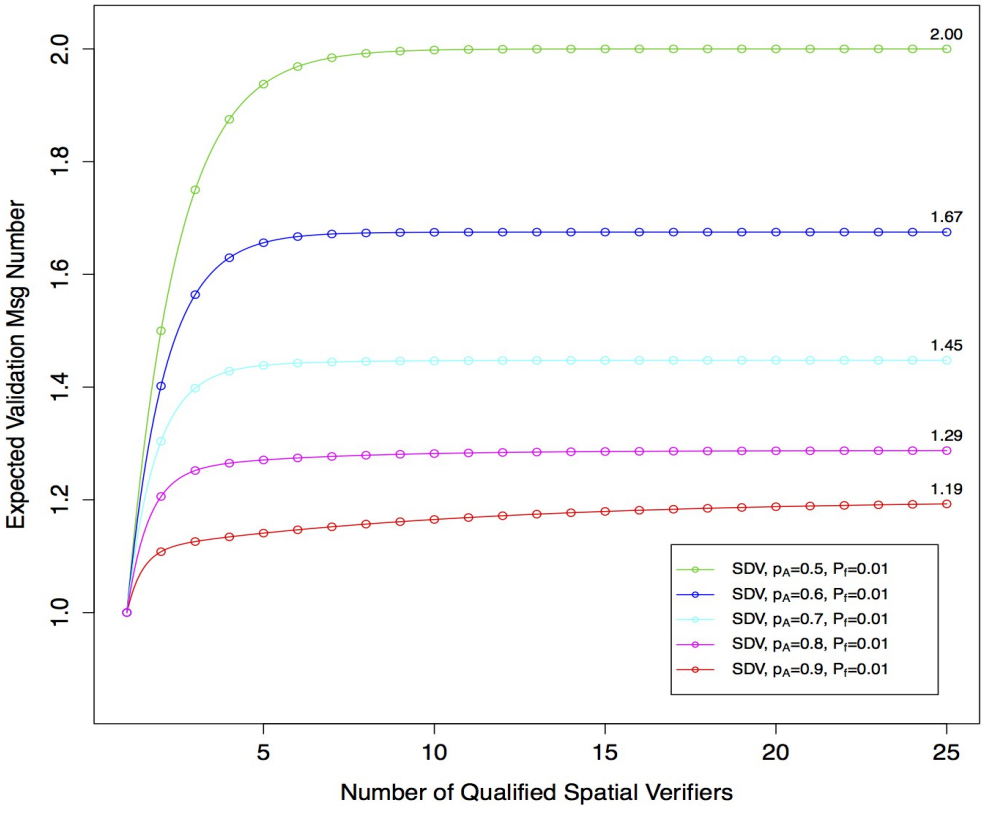
Overhead analysis

- Theoretical

$$\mathbb{E}[N] = \frac{1 - p_A^n}{1 - p_A} P_f + \frac{1 - q_A^n}{1 - q_A}(1 - P_f).$$

- Simulation
  - 1% fault rate

    Larger rates converge too, but possibly on different values
  - Expect around 2 messages to verify each observation



Expected # of verification msgs sent per data

# of verifiers for a source node

Results – 1

- With injected faults

		SHORT	CONST.	NOISE	DRIFT
CASE 1	Sensitivity	1.0 (±0.0)	1.0 (±0.0)	0.958 (±0.022)	1.0 (±0.0)
	Specificity	1.0 (±0.0)	1.0 (±0.0)	0.999 (±0.0)	1.0 (±0.0)
CASE 2	Sensitivity	0.999 (±0.001)	1.0 (±0.001)	0.952 (±0.011)	1.0 (±0.0)
	Specificity	0.996 (±0.005)	0.993 (±0.011)	0.964 (±0.045)	0.995 (±0.009)

High true positive rate

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{FP} + \text{TN}}.$$

Low false positive rate

- Compared with others

	Sensitivity	Specificity
SDV	0.932	1.0
PLA (clean training data)	1.0	0.98
PLA	0.184	0.997

With noisy learning data: effect of robust learning

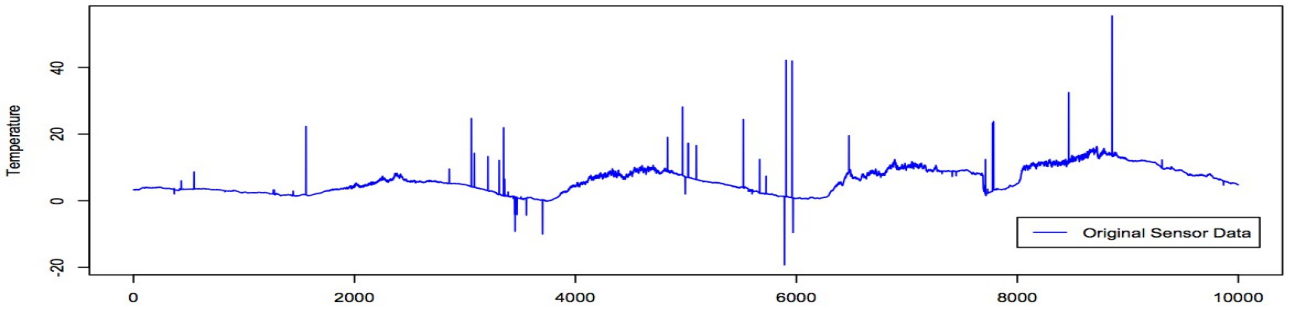
Heuristic; no model formation; not robust; doesn't evolve correlations

Kamal *et alia*. Packet level attestation (PLA): a framework for in-network sensor data reliability. *ACM TOSN*. 2013.

Results – 2

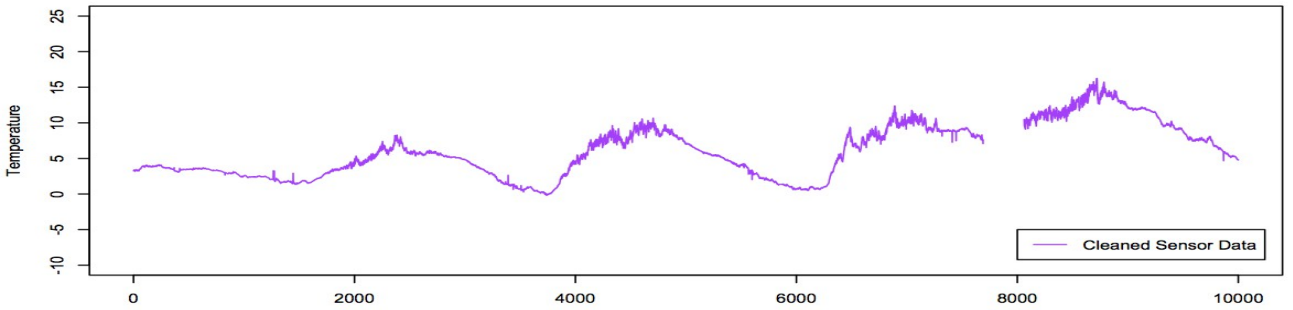
- Against real signals

Original data

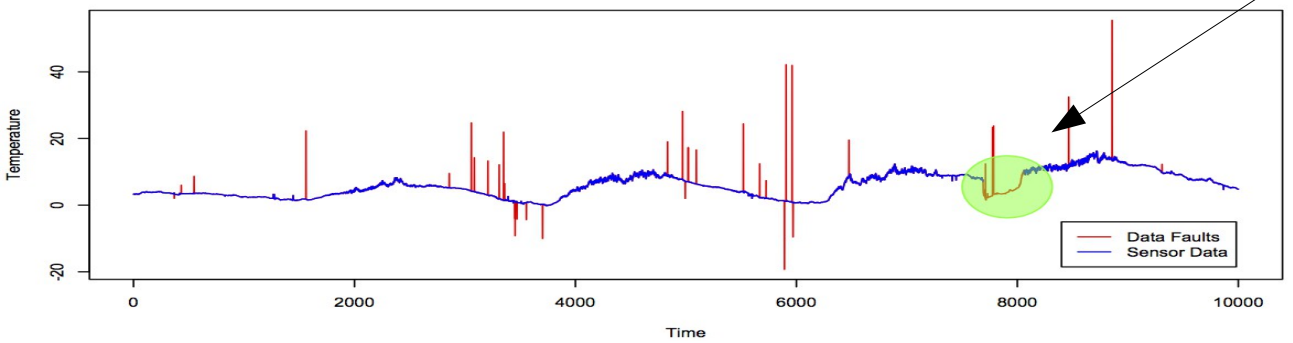


From the Lausanne Urban Canopy Experiment (2006)

Cleaned data



May be a true observation, but not verified by neighbours. Can't tell without ground truth, which of course we don't have



Conclusions

- A more autonomic view of sensor networks
  - Data is a distribution: accept it … and just learn it
  - Scientists typically have the models already
  - But we 're no longer collecting data *per se*

- Techniques aren't as hungry as we might think
  - Comms, memory, and compute all acceptably low
  - Get good fault rejection