Opposing forces in big data

Big data need big model

Einar Holsbø, 261020

Learning outcome: expectation propagation

Rules of thumb

- More data supports richer models: Unconstrained estimation of p numbers requires a sample size N such that you have Kp observations **in the smallest subgroup of your data.** $K=10$ often cited, but known to be optimistic in real life.

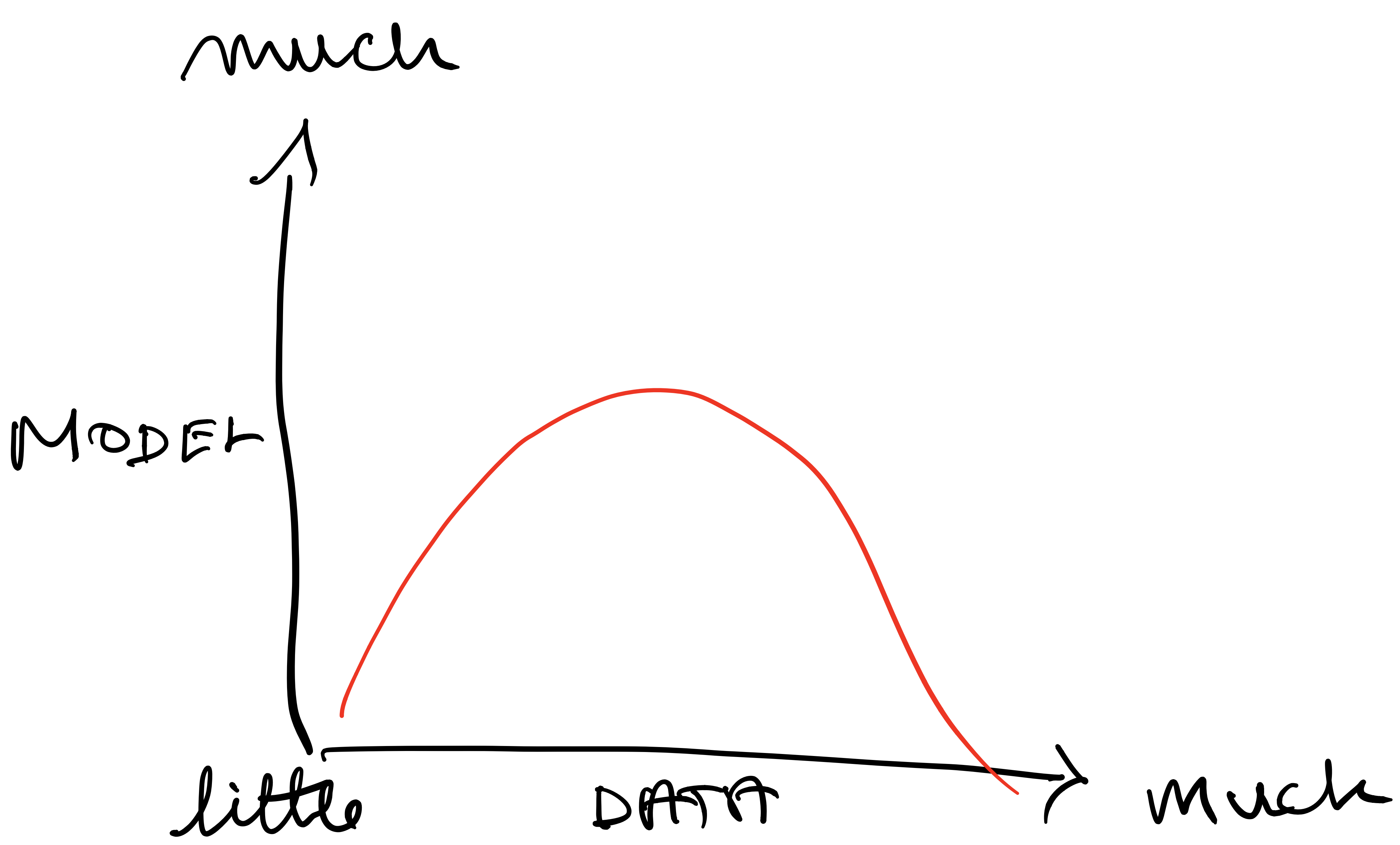
Rules of thumb

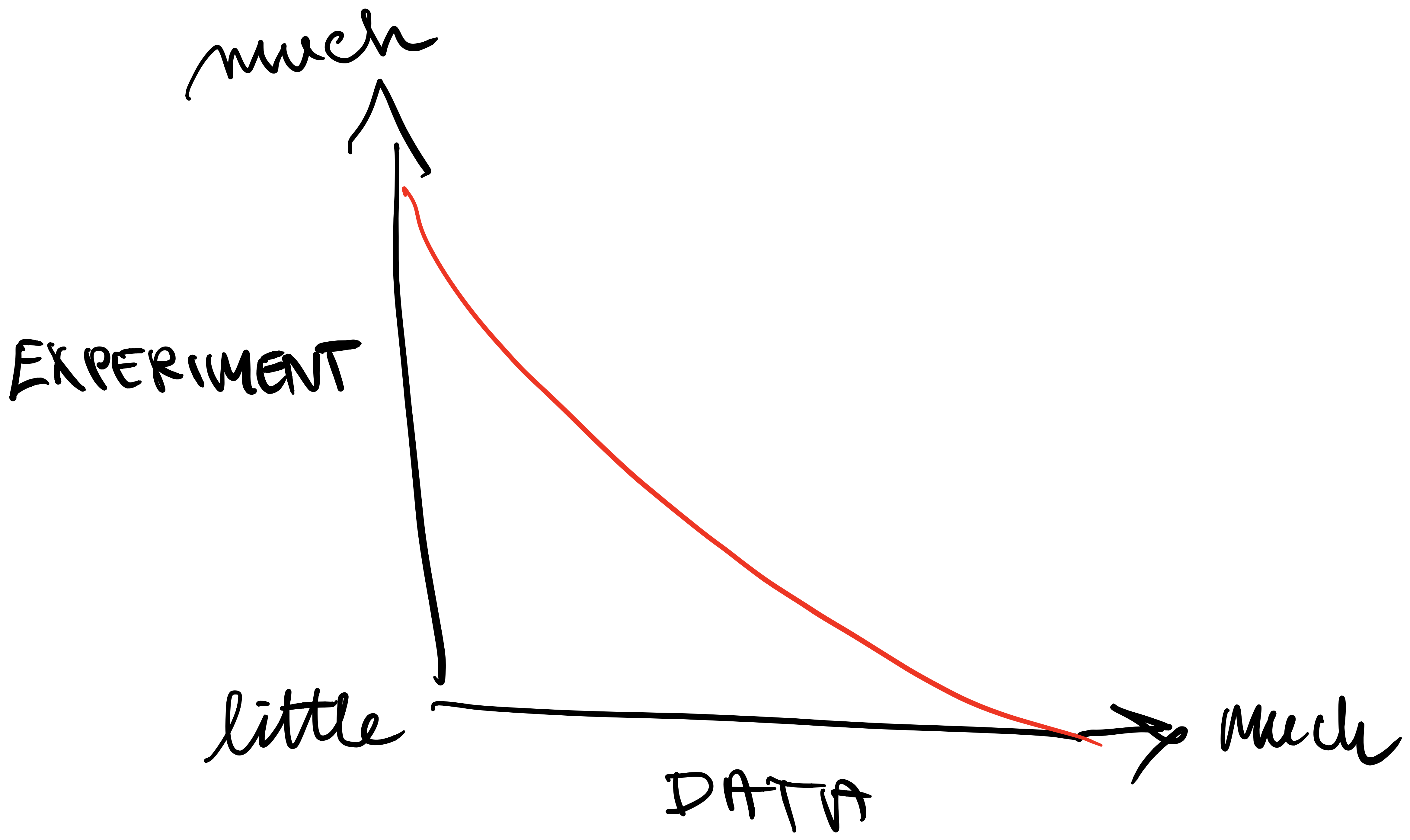
- More data supports richer models: Unconstrained estimation of *p* numbers requires a sample size *N* such that you have *Kp* observations **in the smallest subgroup of your data.** *K=10* often cited, but known to be optimistic in real life.
- More data needs richer models: big data is usually **found data,** so any insights must come from a model that corrects for this. Also, variance in estimation often decreases as 1/sqrt(*N*).

Rules of thumb

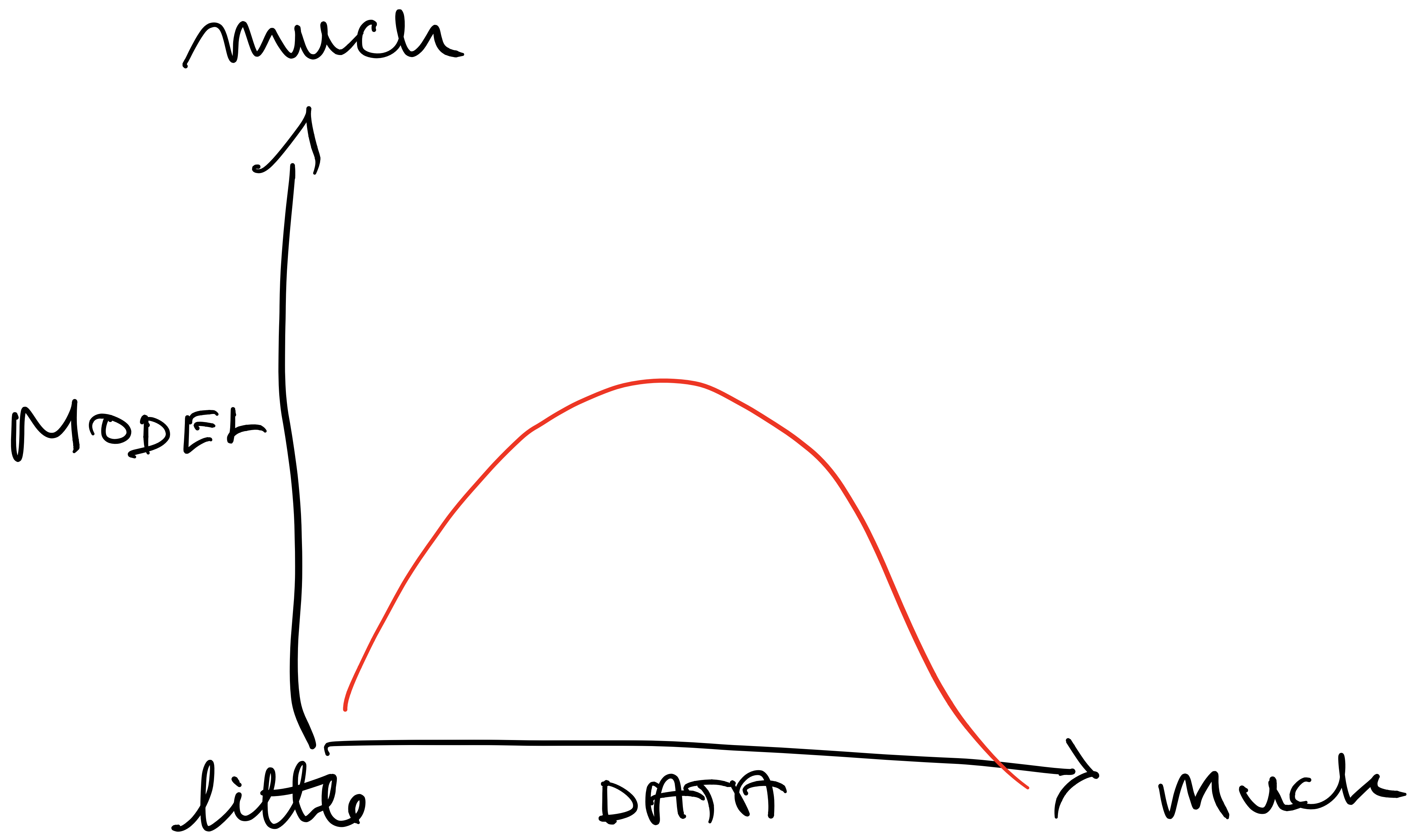
- More data supports richer models: Unconstrained estimation of *p* numbers requires a sample size *N* such that you have *Kp* observations **in the smallest subgroup of your data.** *K=10* often cited, but known to be optimistic in real life.
- More data needs richer models: big data is usually **found data,** so any insights must come from a model that corrects for this. Also, variance in estimation often decreases as 1/sqrt(*N*).
- More data prohibits richer models: Eg. inverting a matrix is 😯(N^3).

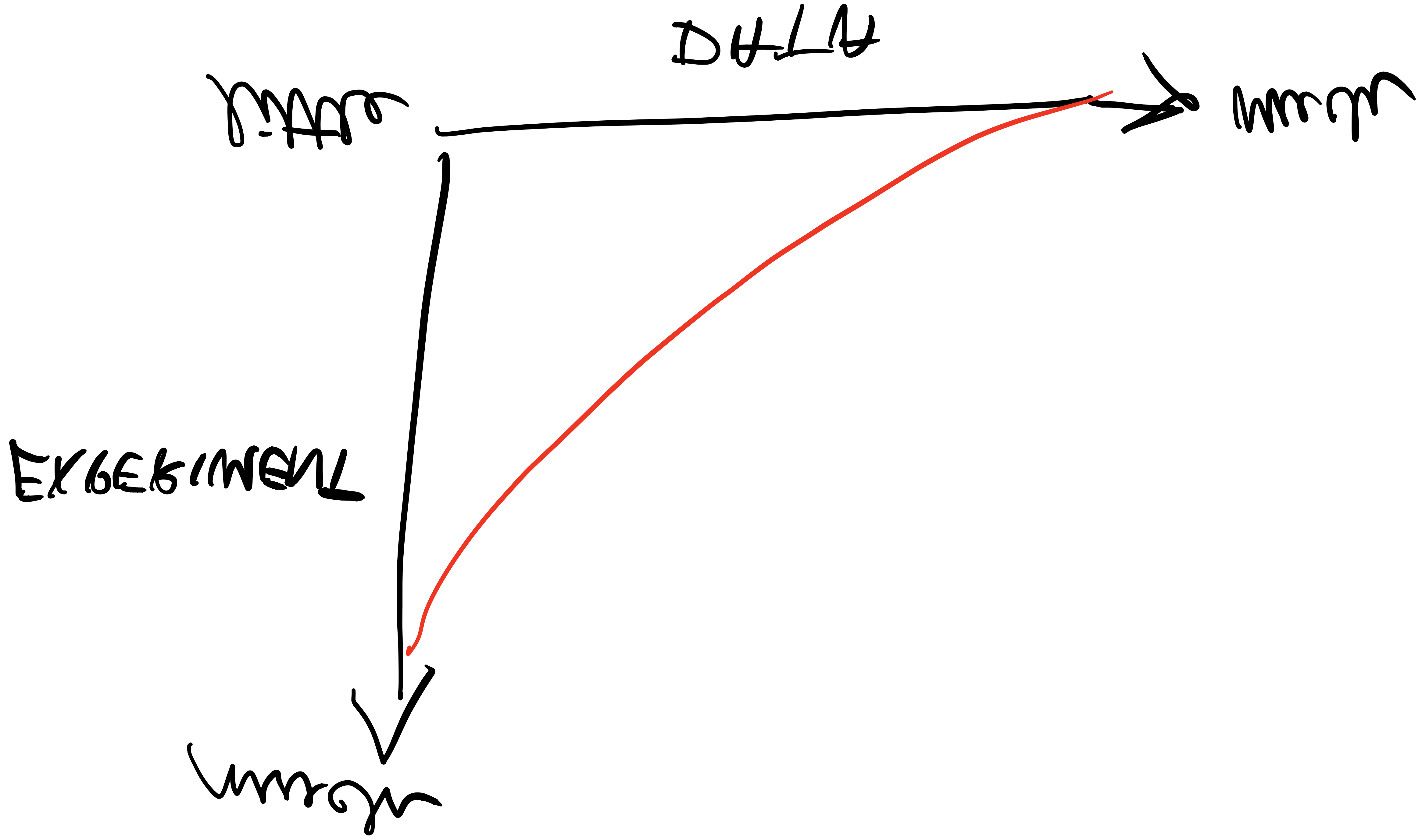
The fundamental problem of Big Data



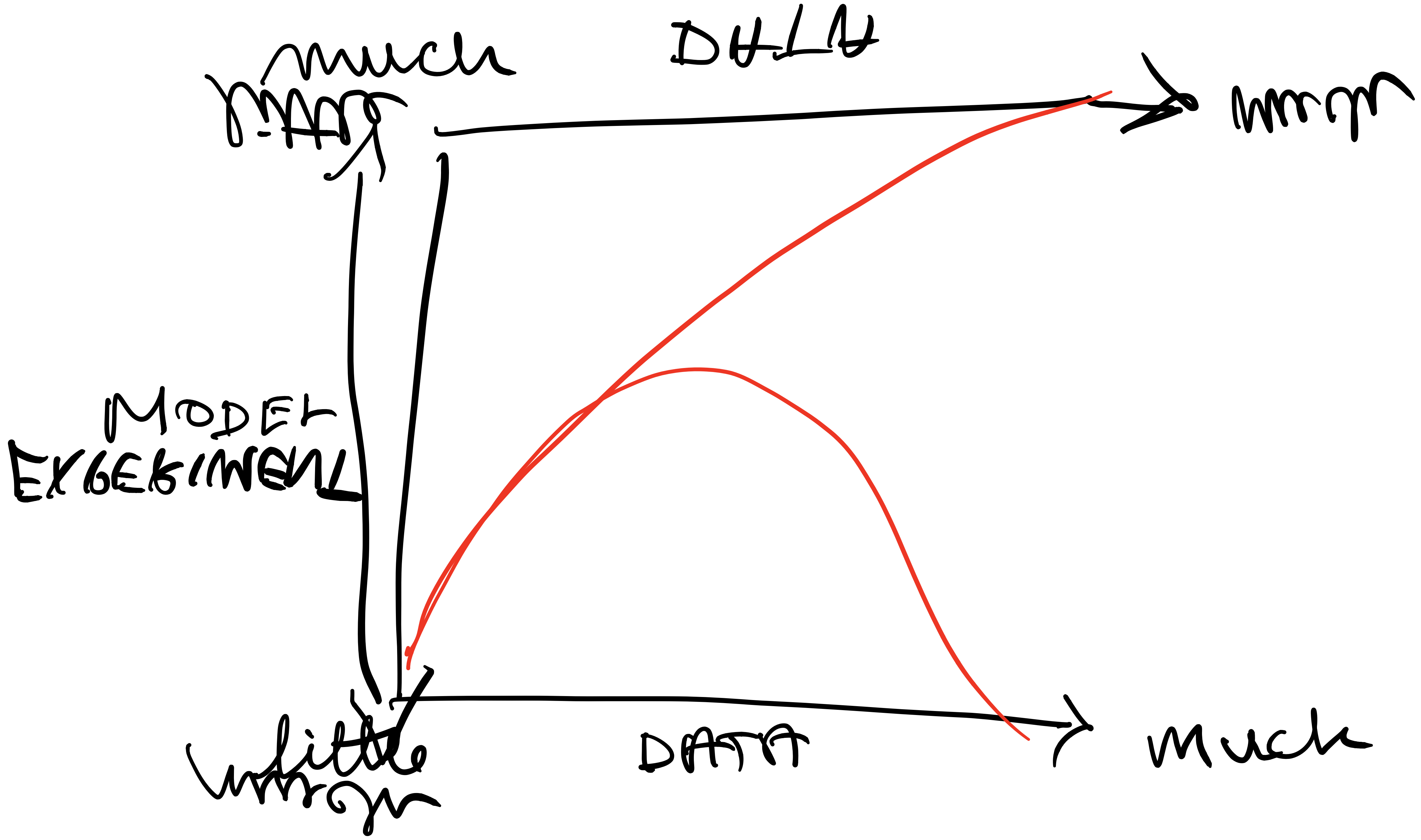


The fundamental problem of Big Data

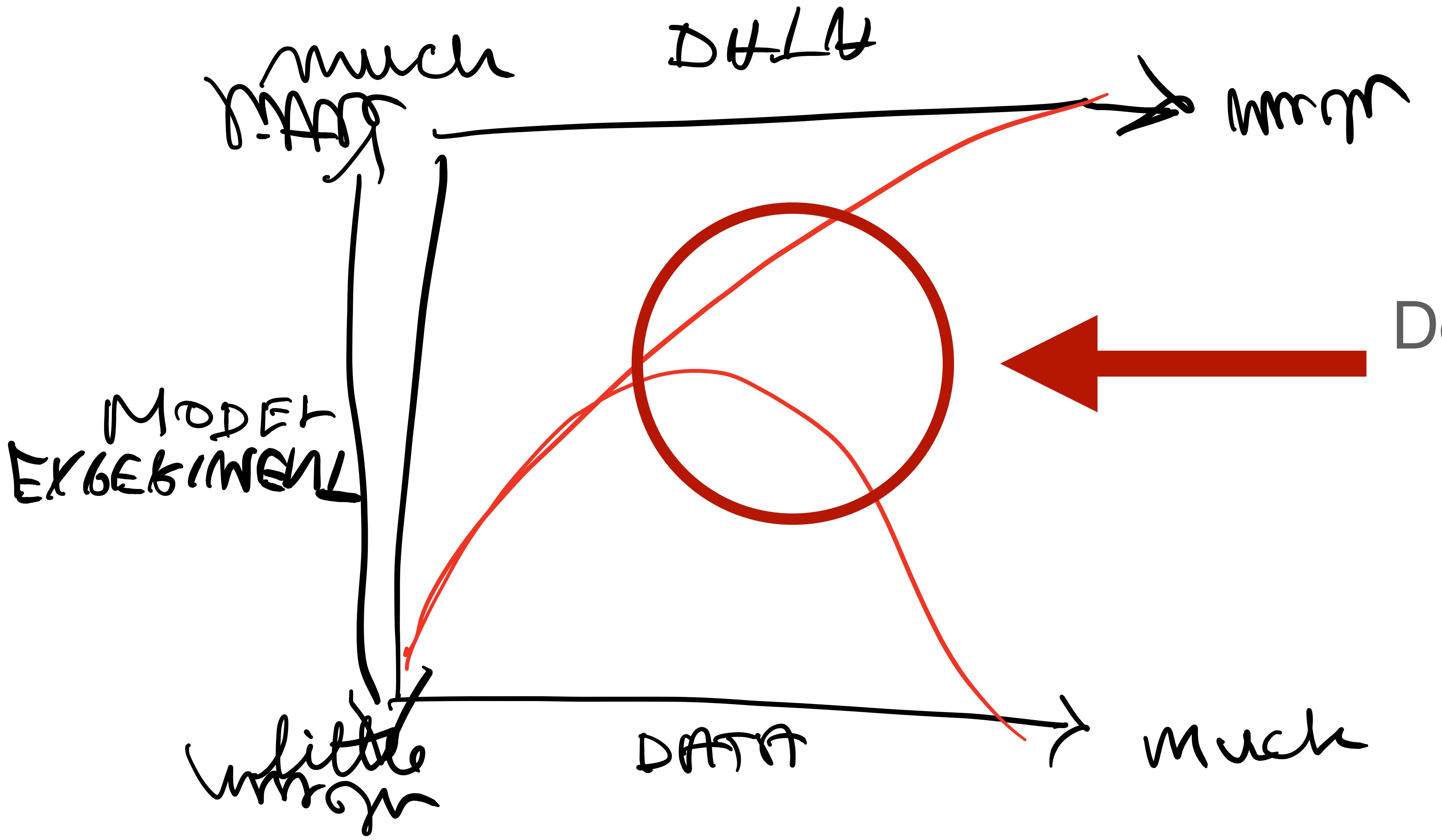




The fundamental problem of Big Data



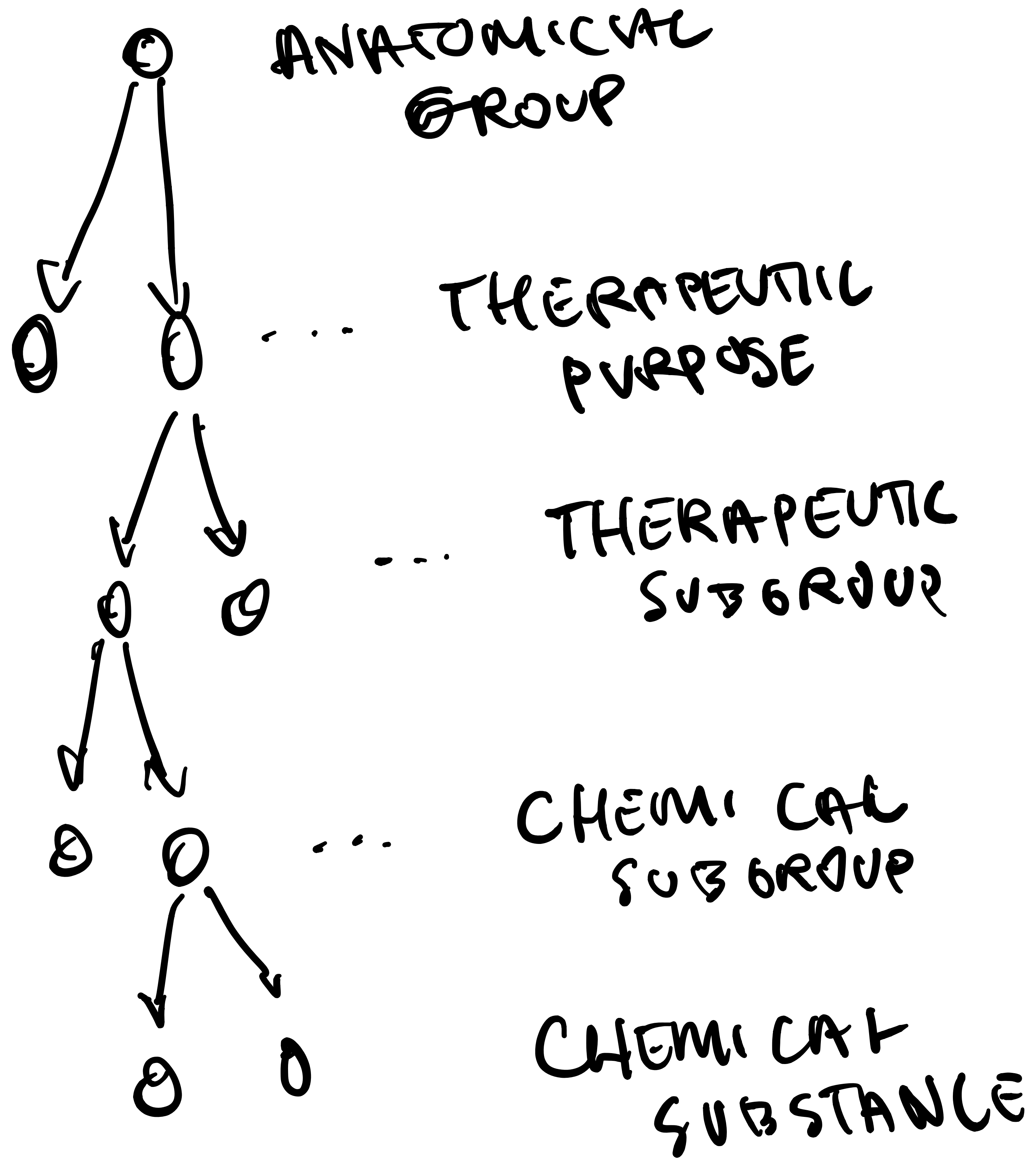
The fundamental problem of Big Data



Demand for model clearly outstripping supply

Motivation: prescription data

- Observational
- Big N
- Rare events
- Hierarchical



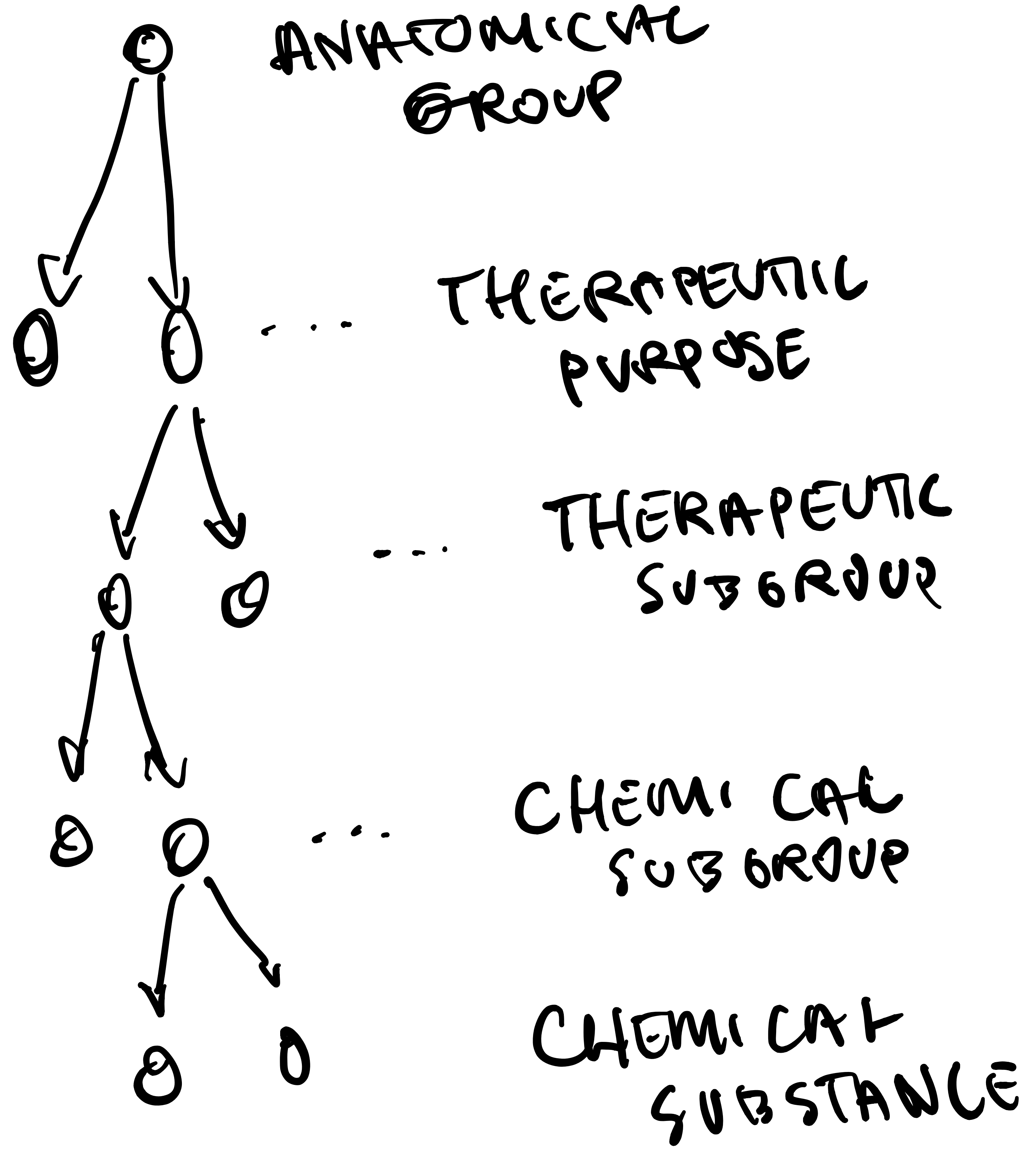
Motivation: prescription data

Rate of adverse events for **chemical substances** probably similar within **chemical subgroup**

$$\log \theta_{\text{SUBSTANCE}} \sim \text{Normal}(\theta_{\text{SUBGROUP}}, \sigma_{\text{SUB.}}),$$

$$\theta_{\text{SUBGROUP}} \sim \text{Normal}(\theta_{\text{THERAPY}}, \sigma_{\text{TH.}}),$$

&c., &c., &c.



Journal of Machine Learning Research 21 (2020) 1-53 Submitted 12/18; Revised 11/19; Published 1/20

Expectation Propagation as a Way of Life: A Framework for Bayesian Inference on Partitioned Data

Aki Vehtari AKI.VEHTARI@AALTO.FI
Department of Computer Science
Aalto University
00076 Aalto, Finland

Andrew Gelman GELMAN@STAT.COLUMBIA.EDU
Departments of Statistics and Political Science
Columbia University
New York, NY 10027, USA

Tuomas Sivula TUOMAS.SIVULA@AALTO.FI

Pasi Jylänki PASI.JYLANKI@GMAIL.COM

Dustin Tran DUSTINVIETTRAN@GMAIL.COM

Swupnil Sahai SWUPNIL@STAT.COLUMBIA.EDU

Paul Blomstedt PAUL.BLOMSTEDT@AALTO.FI

John P. Cunningham JPC2181@COLUMBIA.EDU

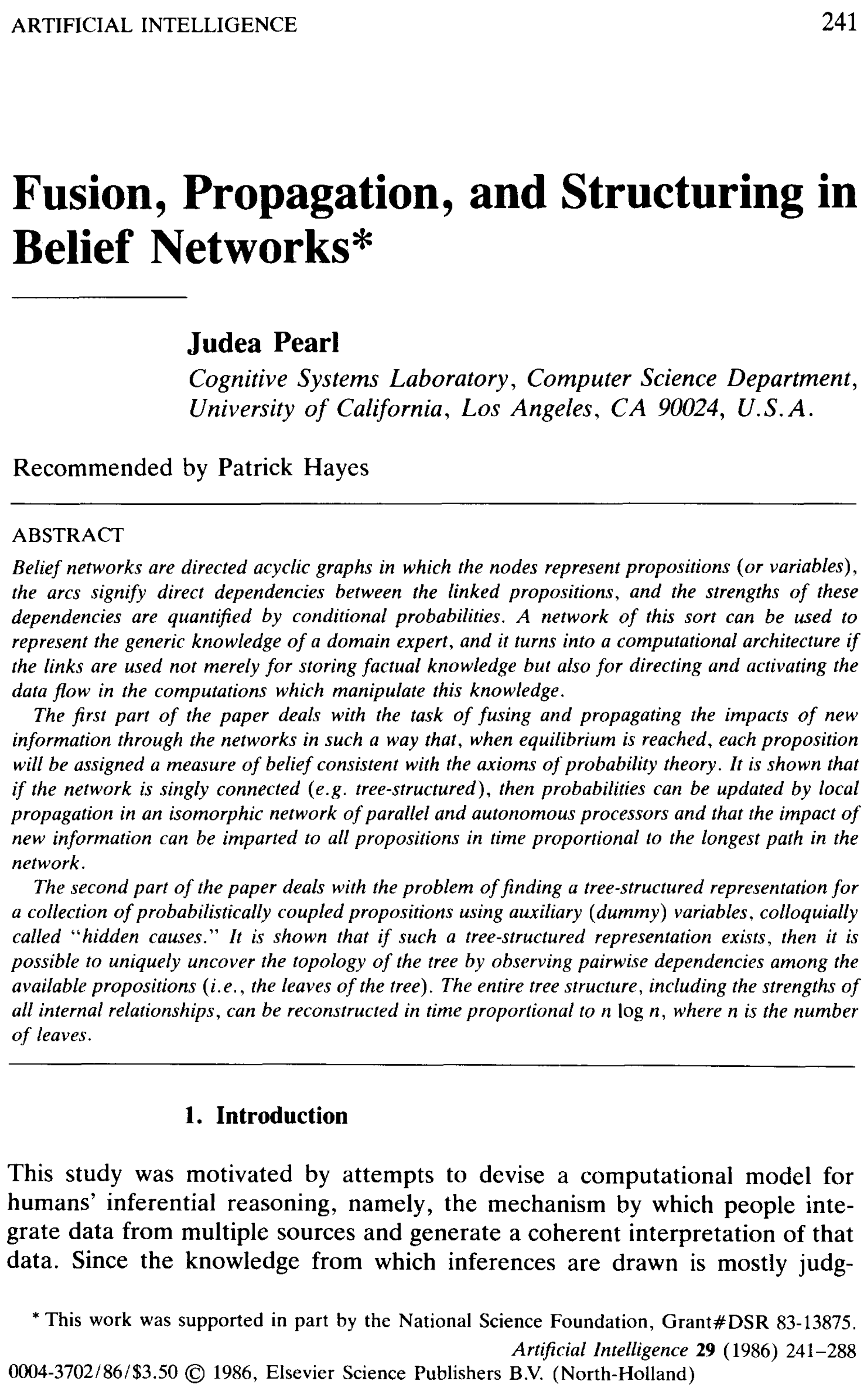
David Schiminovich DS@ASTRO.COLUMBIA.EDU

Christian P. Robert XIAN@CEREMADE.DAUPHINE.FR

Editor: Manfred Opper

- Expectation Propagation algorithm fairly old, presented in Thomas Minka’s PhD dissertation in 2001

- Expectation Propagation algorithm fairly old, presented in Thomas Minka's PhD dissertation in 2001
- The target computation (details to follow) can be seen as sending messages along a **factor graph**


ARTIFICIAL INTELLIGENCE 241

Fusion, Propagation, and Structuring in Belief Networks*

Judea Pearl
Cognitive Systems Laboratory, Computer Science Department, University of California, Los Angeles, CA 90024, U.S.A.

Recommended by Patrick Hayes

ABSTRACT

Belief networks are directed acyclic graphs in which the nodes represent propositions (or variables), the arcs signify direct dependencies between the linked propositions, and the strengths of these dependencies are quantified by conditional probabilities. A network of this sort can be used to represent the generic knowledge of a domain expert, and it turns into a computational architecture if the links are used not merely for storing factual knowledge but also for directing and activating the data flow in the computations which manipulate this knowledge.

The first part of the paper deals with the task of fusing and propagating the impacts of new information through the networks in such a way that, when equilibrium is reached, each proposition will be assigned a measure of belief consistent with the axioms of probability theory. It is shown that if the network is singly connected (e.g. tree-structured), then probabilities can be updated by local propagation in an isomorphic network of parallel and autonomous processors and that the impact of new information can be imparted to all propositions in time proportional to the longest path in the network.

The second part of the paper deals with the problem of finding a tree-structured representation for a collection of probabilistically coupled propositions using auxiliary (dummy) variables, colloquially called "hidden causes." It is shown that if such a tree-structured representation exists, then it is possible to uniquely uncover the topology of the tree by observing pairwise dependencies among the available propositions (i.e., the leaves of the tree). The entire tree structure, including the strengths of all internal relationships, can be reconstructed in time proportional to n log n, where n is the number of leaves.

1. Introduction

This study was motivated by attempts to devise a computational model for humans' inferential reasoning, namely, the mechanism by which people integrate data from multiple sources and generate a coherent interpretation of that data. Since the knowledge from which inferences are drawn is mostly judg-

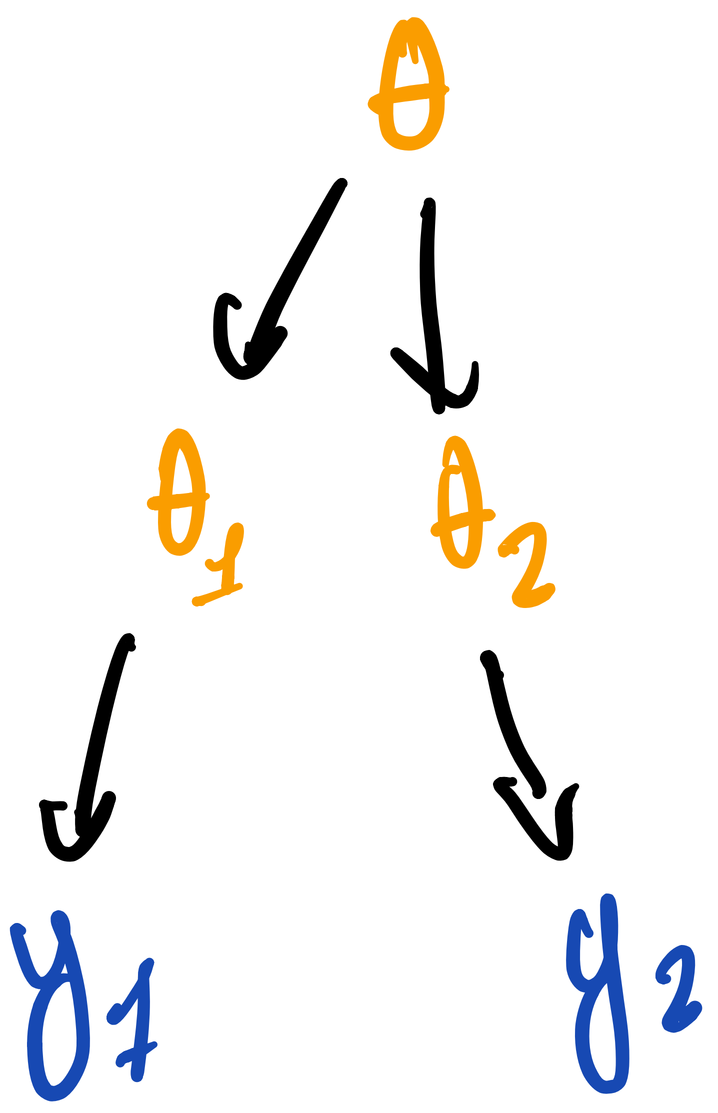
* This work was supported in part by the National Science Foundation, Grant#DSR 83-13875.

Artificial Intelligence **29** (1986) 241–288
0004-3702/86/$3.50 © 1986, Elsevier Science Publishers B.V. (North-Holland)

- Expectation Propagation algorithm fairly old, presented in Thomas Minka's PhD dissertation in 2001
- The target computation (details to follow) can be seen as sending messages along a **factor graph**
- Message passing idea traces back to Judea Pearl in 1986

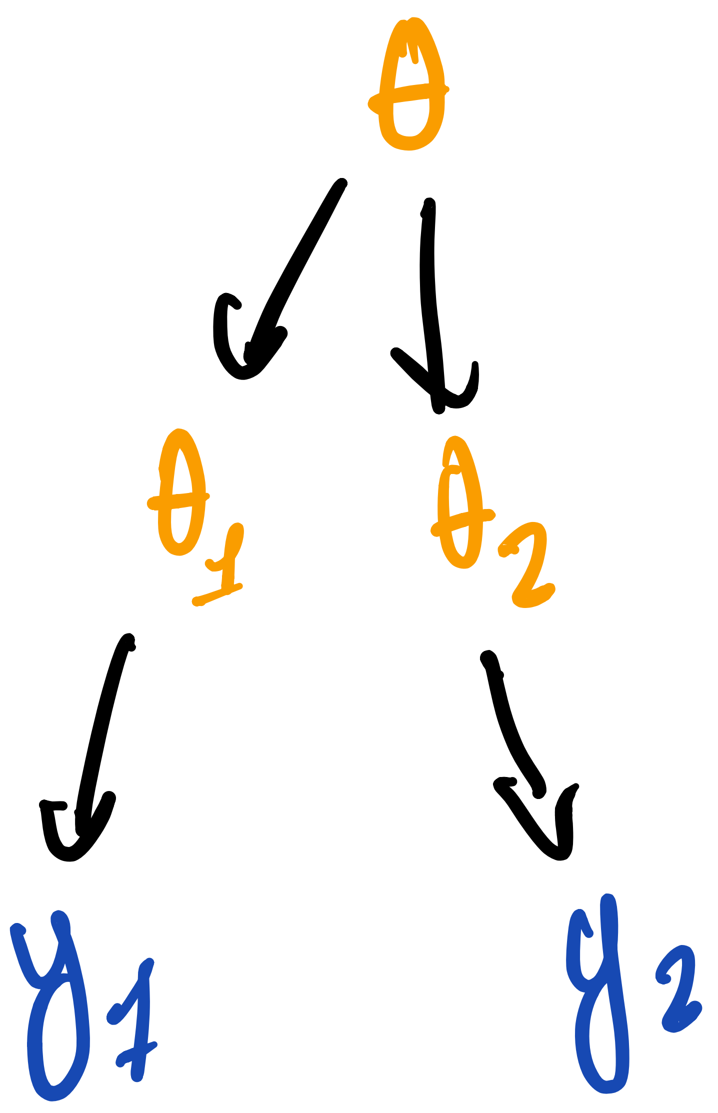
Factor graphs

TOY MODEL
4 DRUG DATA:



Factor graphs

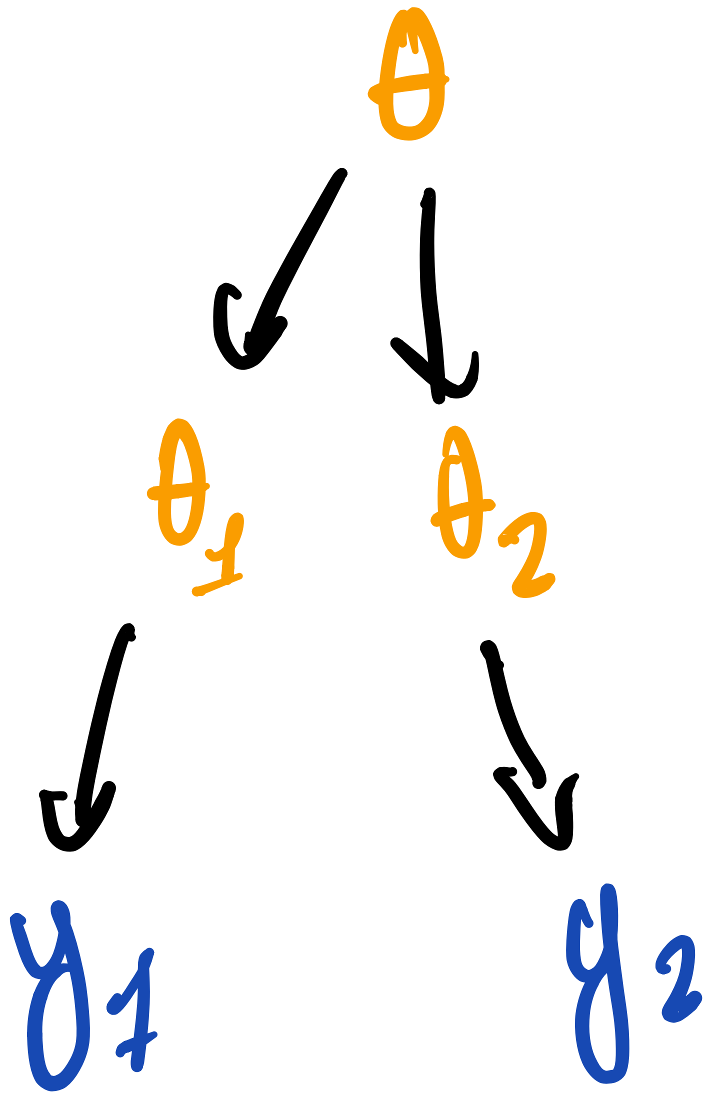
TOY MODEL
4 DRUG DATA:



$$f(\theta, \theta_1, \theta_2 \mid y_1, y_2)$$

Factor graphs

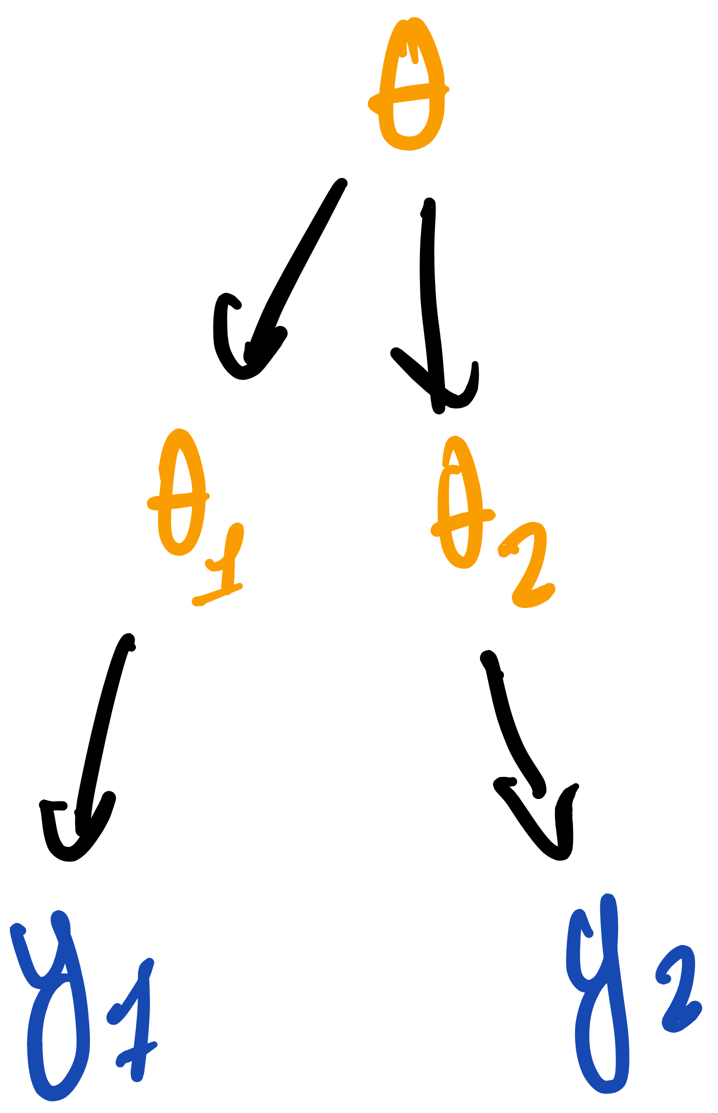
TOY MODEL
4 DRUG DATA:



$$f(\theta, \theta_1, \theta_2 \mid y_1, y_2) \propto f(y_1, y_2 \mid \theta, \theta_1, \theta_2)\, f(\theta, \theta_1, \theta_2)$$

Factor graphs

TOY MODEL
4 DRUG DATA:



$$f(\theta, \theta_1, \theta_2 \mid y_1, y_2) \propto f(y_1, y_2 \mid \theta, \theta_1, \theta_2)\, f(\theta, \theta_1, \theta_2)$$

$$= f(y_1 \mid \theta_1)\, f(y_2 \mid \theta_2)\, f(\theta_1 \mid \theta)\, f(\theta_2 \mid \theta)\, f(\theta)$$

Factor graphs

$$\underbrace{f(y_1|\theta_1)}_{f_1}\underbrace{f(y_2|\theta_2)}_{f_2}\underbrace{f(\theta_1|\theta)}_{f_3}\underbrace{f(\theta_2|\theta)}_{f_4}\underbrace{f(\theta)}_{f_5}$$

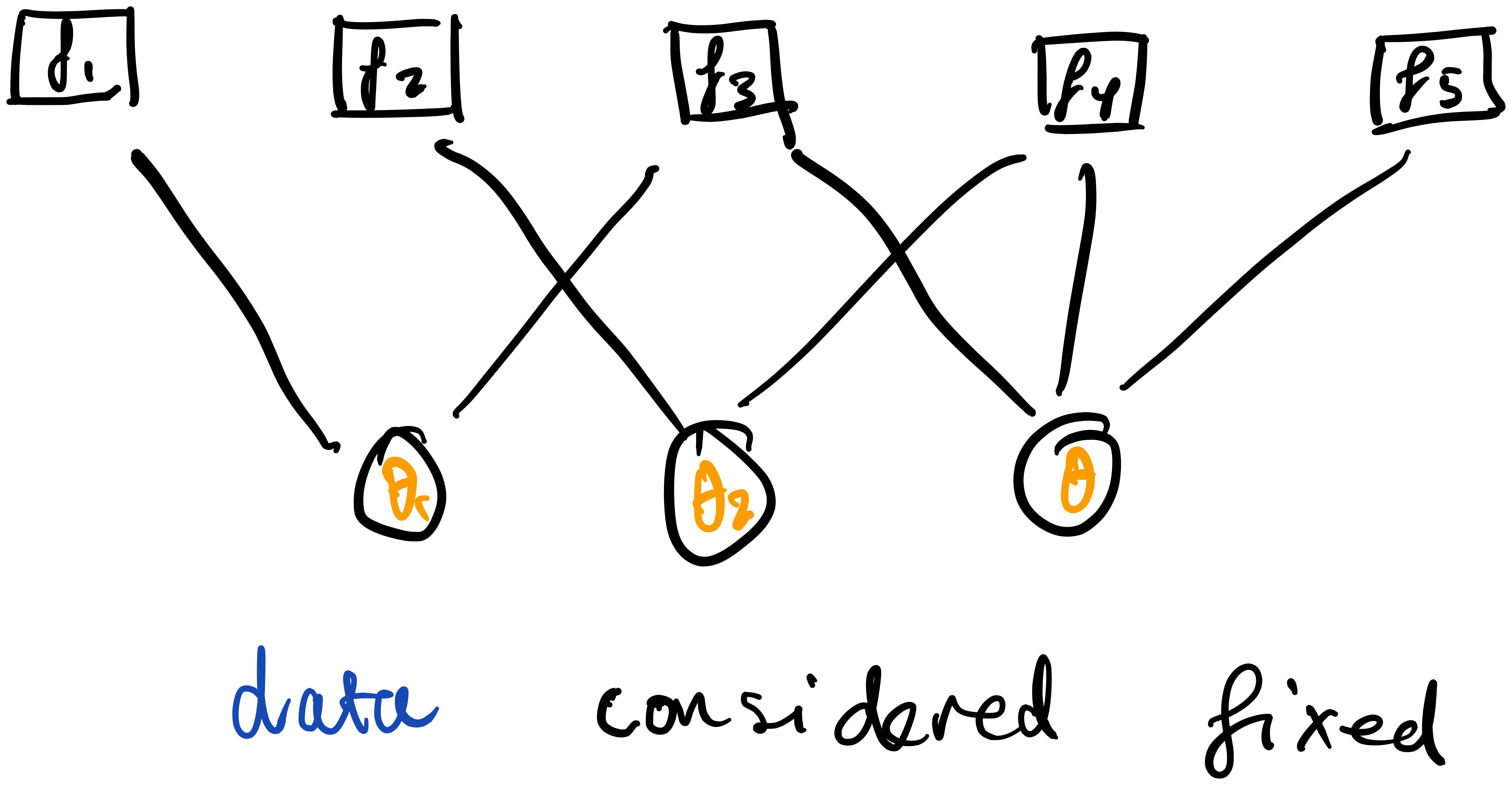
Factor graphs

$$\underbrace{f(y_1 \mid \theta_1)}_{f_1}\,\underbrace{f(y_2 \mid \theta_2)}_{f_2}\,\underbrace{f(\theta_1 \mid \theta)}_{f_3}\,\underbrace{f(\theta_2 \mid \theta)}_{f_4}\,\underbrace{f(\theta)}_{f_5}$$

$\boxed{f_1}$ $\boxed{f_2}$ $\boxed{f_3}$ $\boxed{f_4}$ $\boxed{f_5}$

Factor graphs

$$\underbrace{f(y_1 \mid \theta_1)}_{f_1}\underbrace{f(y_2 \mid \theta_2)}_{f_2}\underbrace{f(\theta_1 \mid \theta)}_{f_3}\underbrace{f(\theta_2 \mid \theta)}_{f_4}\underbrace{f(\theta)}_{f_5}$$

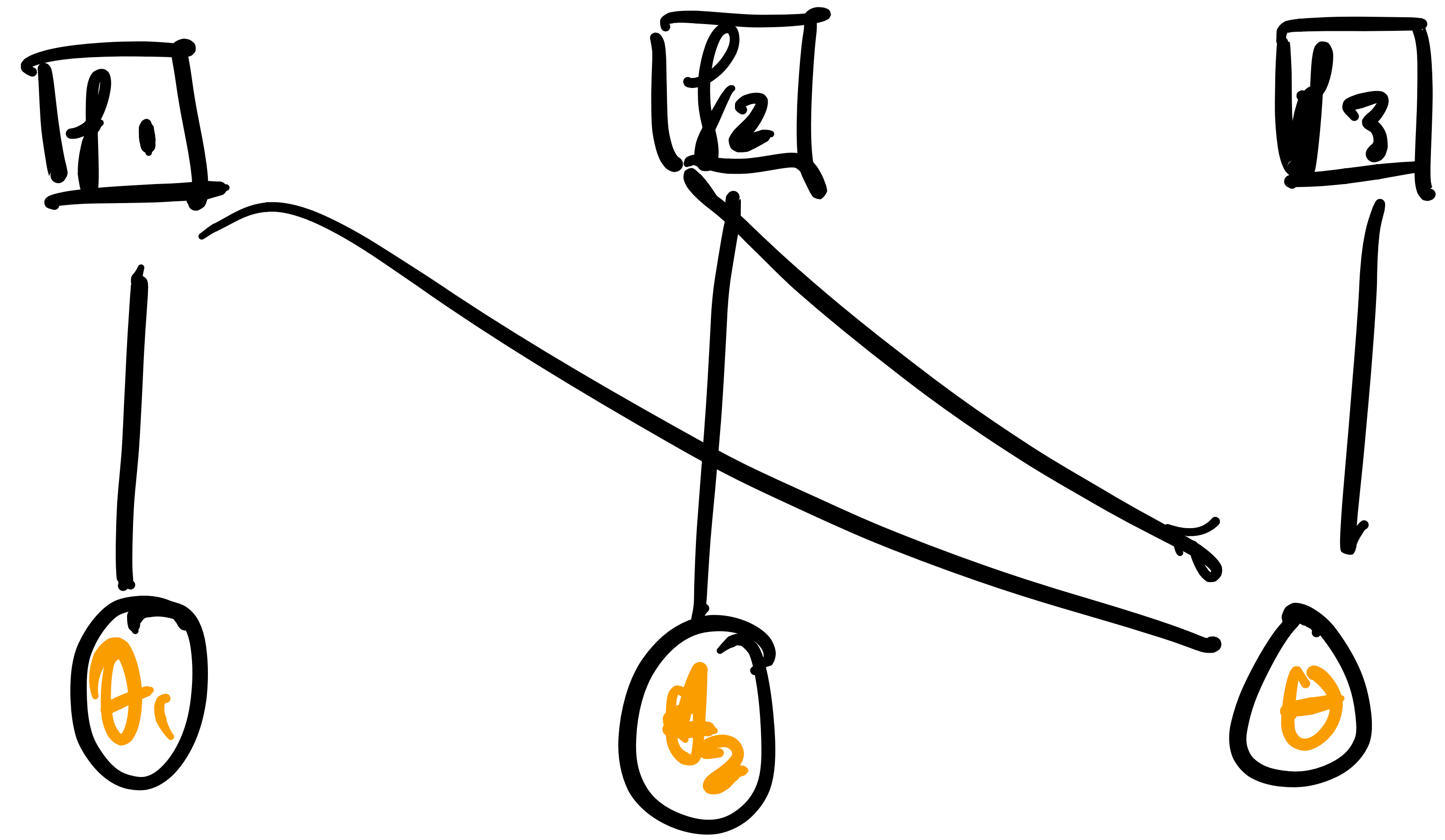


Factorization not unique

$$\underbrace{f(y_1 \mid \theta_1)\, f(\theta_1 \mid \theta)}_{f_1}\ \underbrace{f(y_2 \mid \theta_2)\, f(\theta_2 \mid \theta)}_{f_2}\ \underbrace{f(\theta)}_{f_3}$$

Factorization not unique

$$\underbrace{f(y_1 \mid \theta_1)\, f(\theta_1 \mid \theta)}_{f_1}\; \underbrace{f(y_2 \mid \theta_2)\, f(\theta_2 \mid \theta)}_{f_2}\; \underbrace{f(\theta)}_{f_3}$$



Factorization not unique

Partitioning along data perhaps particularly simple (always possible with the usual iid assumptions)

Factorization not unique

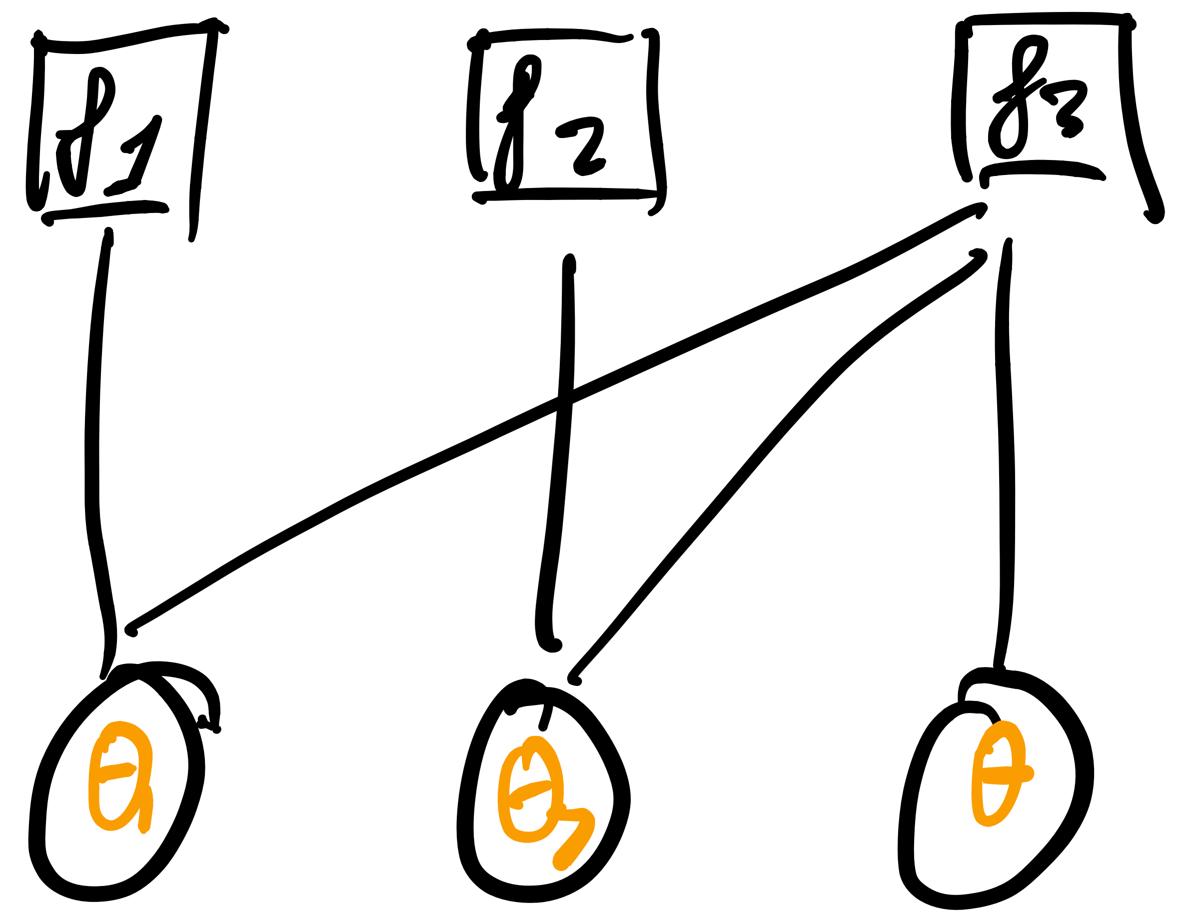
$$\underbrace{f(y_1 \mid \theta_1)}_{f_1}\underbrace{f(y_2 \mid \theta_2)}_{f_2}\underbrace{f(\theta_1 \mid \theta) f(\theta_2 \mid \theta) f(\theta)}_{f_3}$$

Partitioning along data perhaps particularly simple (always possible with the usual iid assumptions)

Factorization not unique

$$\underbrace{f(y_1|\theta_1)}_{f_1}\underbrace{f(y_2|\theta_2)}_{f_2}\underbrace{f(\theta_1|\theta)f(\theta_2|\theta)\,f(\theta)}_{f_3}$$

Partitioning along data perhaps particularly simple (always possible with the usual iid assumptions)



Factorization not unique

$$\underbrace{f(y_1|\theta_1)}_{f_1}\underbrace{f(y_2|\theta_2)}_{f_2}\underbrace{f(\theta_1|\theta)f(\theta_2|\theta)f(\theta)}_{f_3}$$

Partitioning along data perhaps particularly simple (always possible with the usual iid assumptions)

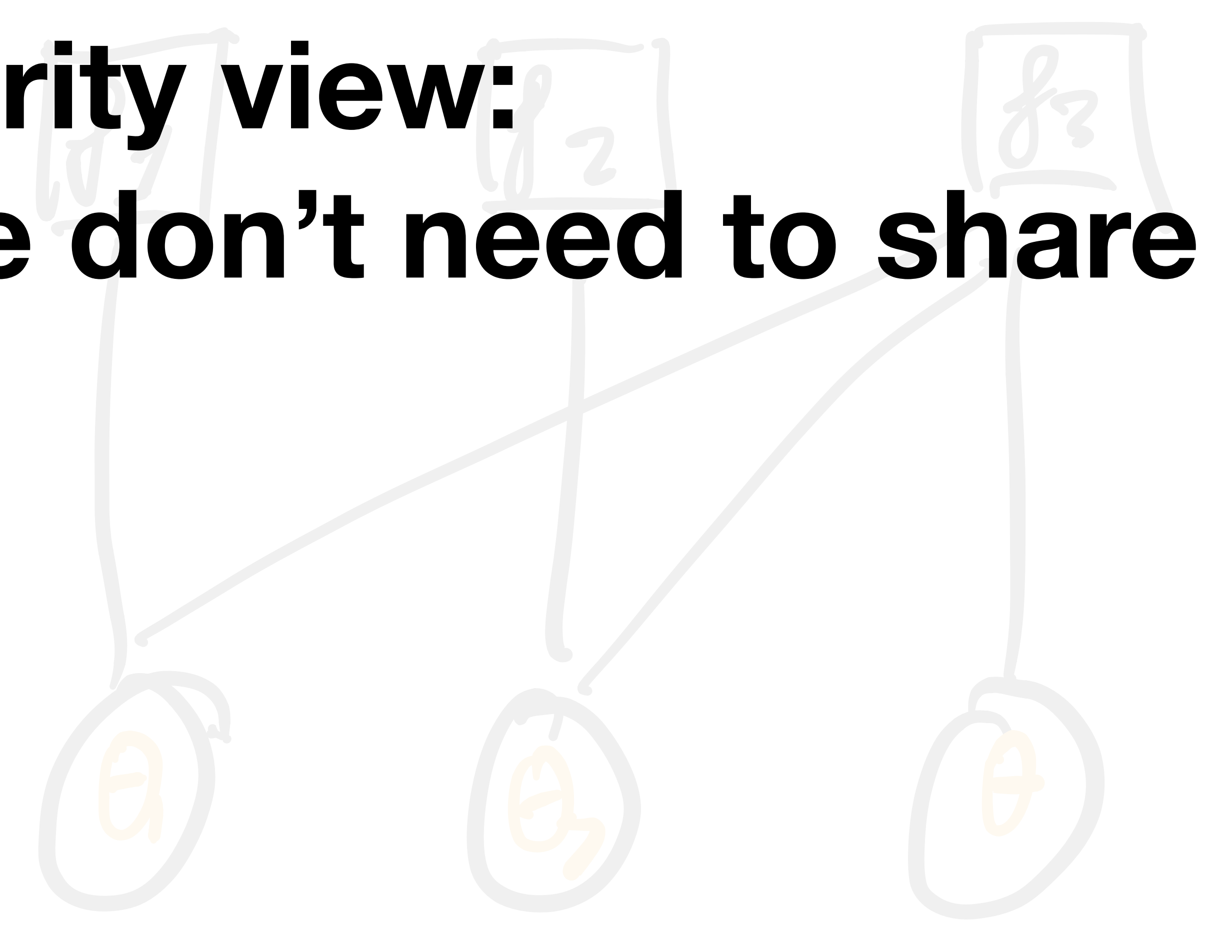
Top-down computational view:
nice because we get so split up our Big Data

Factorization not unique

Partitioning along data perhaps particularly simple (always possible with the usual iid assumptions)

Top-down computational view:
nice because we get so split up our Big Data

Bottom-up security view:
nice because we don't need to share our Secret Data



Goal: approximate full function by approximating at the “sites” f_i passing θ values along edges in the graph iteratively

$$f(\underline{\theta}) = \prod_{k=0}^{K} f_k(\underline{\theta})$$

$$f(\underline{\theta}) = \prod_{k=0}^{K} f_k(\underline{\theta}) \quad \text{approx. w:} \quad \prod_{k=0}^{K} g_k(\underline{\theta}) = g(\underline{\theta})$$

$$f(\underline{\theta}) = \prod_{k=0}^{K} f_k(\underline{\theta}) \quad \text{approx. w:} \quad \underbrace{\prod_{k=0}^{K} g_k(\underline{\theta}) = g(\underline{\theta})}_{\text{probably Gaussian}}$$

$$f(\underline{\theta}) = \prod_{k=0}^{K} f_k(\underline{\theta})$$

approx. w:

$$\underbrace{\prod_{k=0}^{K} g_k(\underline{\theta}) = g(\underline{\theta})}_{\text{probably Gaussian}}$$

EXPECTATION PROPAGATION

At each site f_i:

$$f(\underline{\theta}) = \prod_{k=0}^{K} f_k(\underline{\theta}) \quad \text{approx. w:} \quad \underbrace{\prod_{k=0}^{K} g_k(\underline{\theta}) = g(\underline{\theta})}_{\text{probably Gaussian}}$$

EXPECTATION PROPAGATION

At each site f_i:

1. Form $g_{-i}(\underline{\theta}) = \dfrac{g(\underline{\theta})}{g_i(\underline{\theta})}$

$$f(\underline{\theta}) = \prod_{k=0}^{K} f_k(\underline{\theta}) \quad \text{approx. w:} \quad \underbrace{\prod_{k=0}^{K} g_k(\underline{\theta}) = g(\underline{\theta})}_{\text{probably Gaussian}}$$

EXPECTATION PROPAGATION

At each site f_i:

1. Form $g_{-i}(\underline{\theta}) = \dfrac{g(\underline{\theta})}{g_i(\underline{\theta})}$

2. Estimate $g_{\backslash i}(\underline{\theta}) = f_i(\underline{\theta})\, g_{-i}(\underline{\theta})$

$$f(\underline{\theta}) = \prod_{k=0}^{K} f_k(\underline{\theta}) \quad \text{approx. w:} \quad \underbrace{\prod_{k=0}^{K} g_k(\underline{\theta}) = g(\underline{\theta})}_{\text{probably Gaussian}}$$

EXPECTATION PROPAGATION

At each site f_i:

1. Form $g_{-i}(\underline{\theta}) = \dfrac{g(\underline{\theta})}{g_i(\underline{\theta})}$

2. Estimate $g_{\backslash i}(\underline{\theta}) = f_i(\underline{\theta})\, g_{-i}(\underline{\theta})$

3. Let $g_{new}(\underline{\theta}) \leftarrow g_{\backslash i}(\underline{\theta})$

$$f(\underline{\theta}) = \prod_{k=0}^{K} f_k(\underline{\theta}) \quad \text{approx. w:} \quad \underbrace{\prod_{k=0}^{K} g_k(\underline{\theta}) = g(\underline{\theta})}_{\text{probably Gaussian}}$$

EXPECTATION PROPAGATION

At each site f_i:

1. Form $g_{-i}(\underline{\theta}) = \dfrac{g(\underline{\theta})}{g_i(\underline{\theta})}$

2. Estimate $g_{\backslash i}(\underline{\theta}) = f_i(\underline{\theta})\, g_{-i}(\underline{\theta})$

3. Let $g_{new}(\underline{\theta}) \leftarrow g_{\backslash i}(\underline{\theta})$

4. Communicate $g_{new}(\underline{\theta})$

$$f(\underline{\theta}) = \prod_{k=0}^{K} f_k(\underline{\theta}) \quad \text{approx. w:} \quad \underbrace{\prod_{k=0}^{K} g_k(\underline{\theta}) = g(\underline{\theta})}_{\text{probably Gaussian}}$$

EXPECTATION PROPAGATION

At each site f_i:

1. Form $g_{-i}(\underline{\theta}) = \dfrac{g(\underline{\theta})}{g_i(\underline{\theta})}$

2. Estimate $g_{\backslash i}(\underline{\theta}) = f_i(\underline{\theta})\, g_{-i}(\underline{\theta})$ ← THE EXPENSIVE PART

3. Let $g_{new}(\underline{\theta}) \leftarrow g_{\backslash i}(\underline{\theta})$

4. Communicate $g_{new}(\underline{\theta})$

Must iterate until convergence; convergence not guaranteed

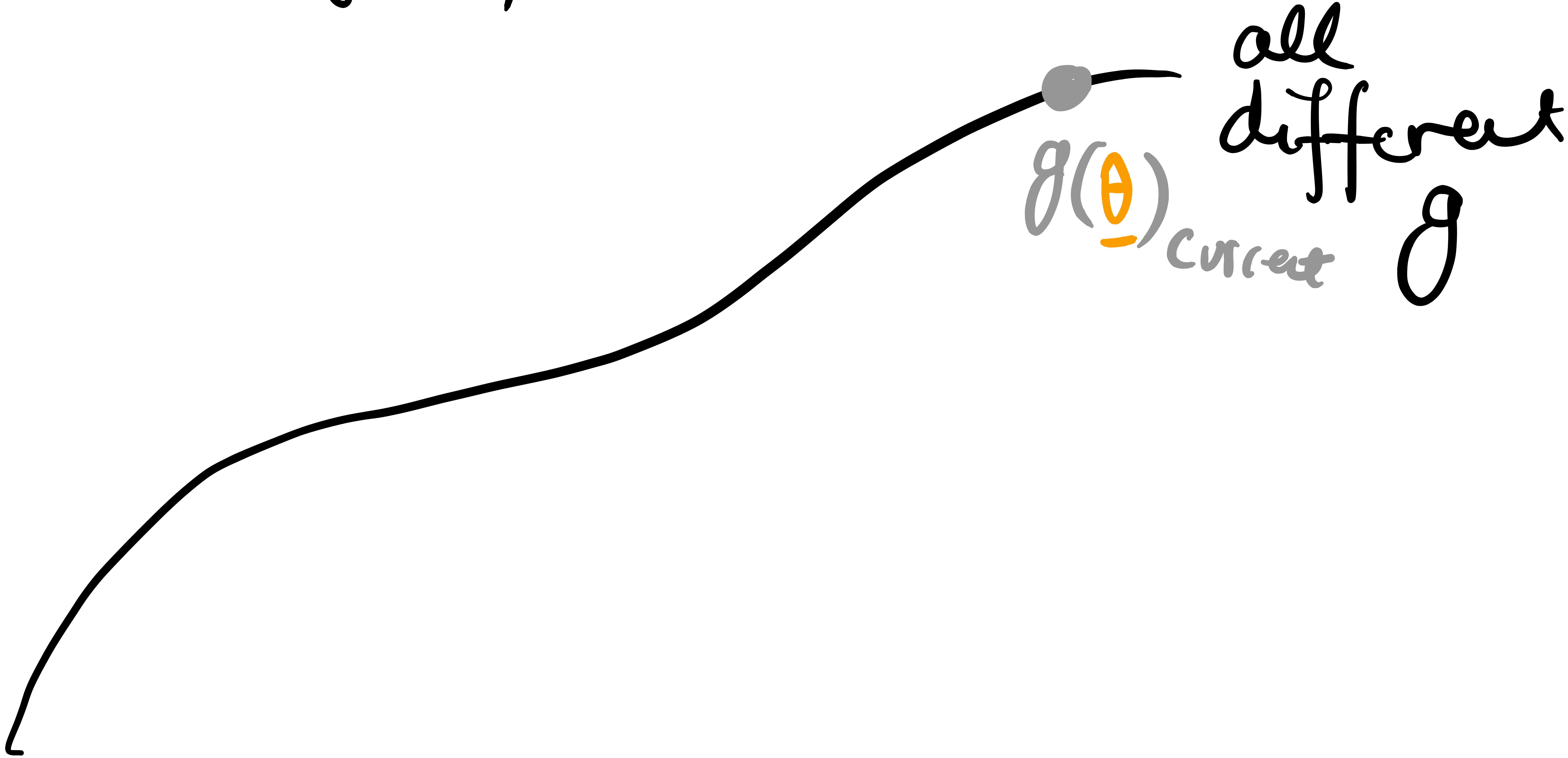
"DISTRIBUTION SPACE"

- $f(\underline{\theta})$

all different g

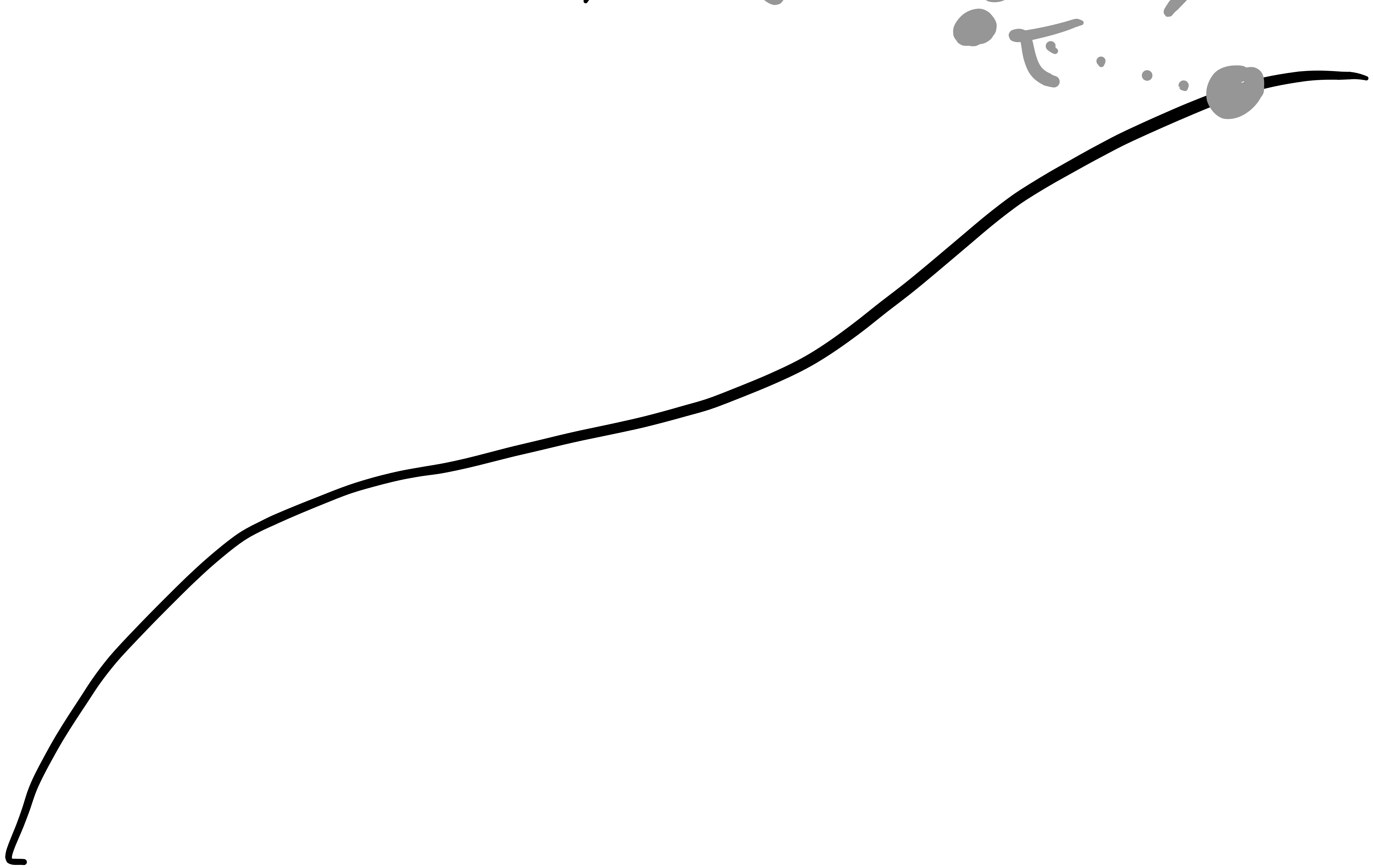
"DISTRIBUTION SPACE"

- $f(\underline{\theta})$



"DISTRIBUTION SPACE"

• $f(\underline{\theta})$ $f_i(\underline{\theta})\, g_{-i}(\underline{\theta})$



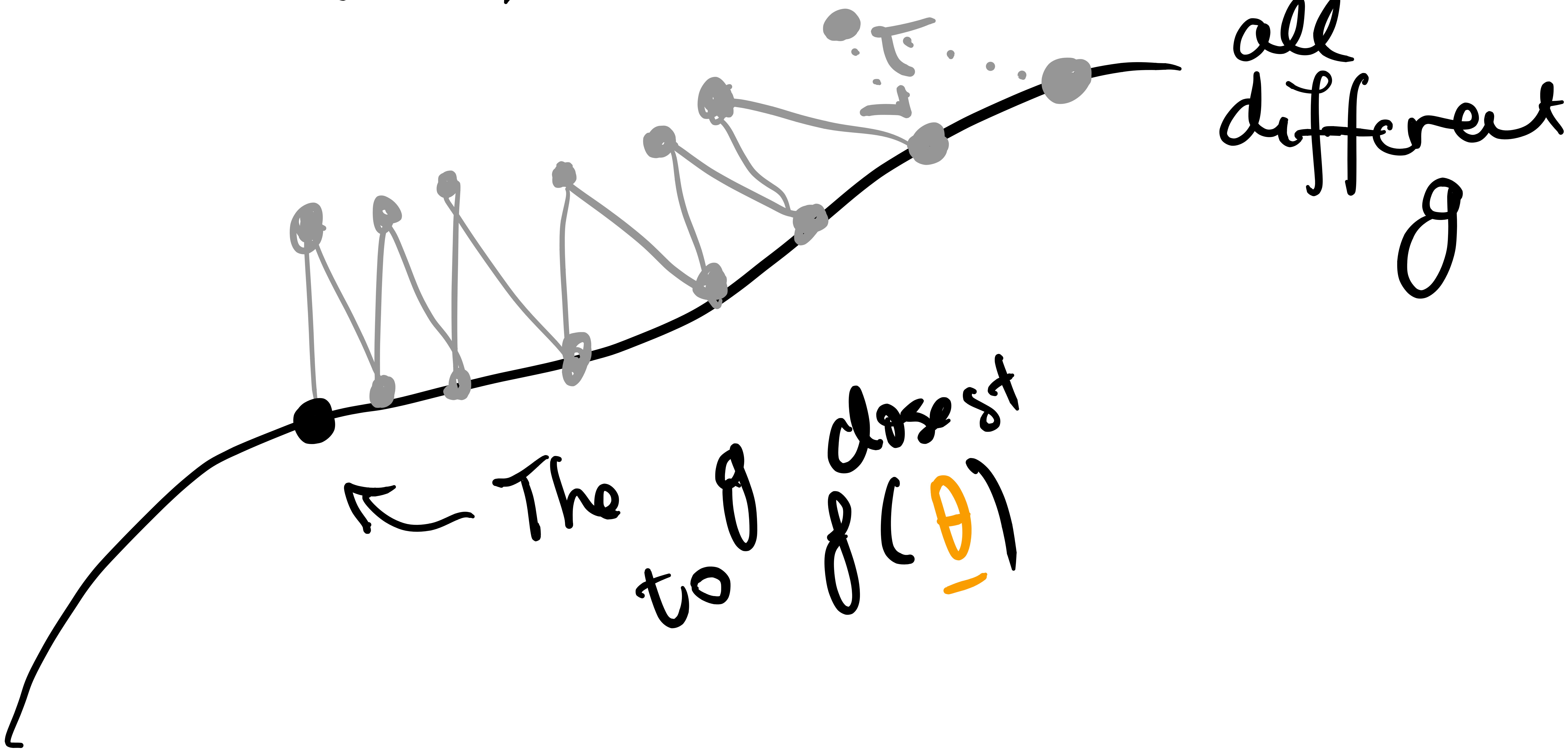
all different g

"DISTRIBUTION SPACE"

- $f(\underline{\theta})$

"DISTRIBUTION SPACE"

• $f(\underline{\theta})$



The current g is needed at all sites

Makes for a simple distributed architecture

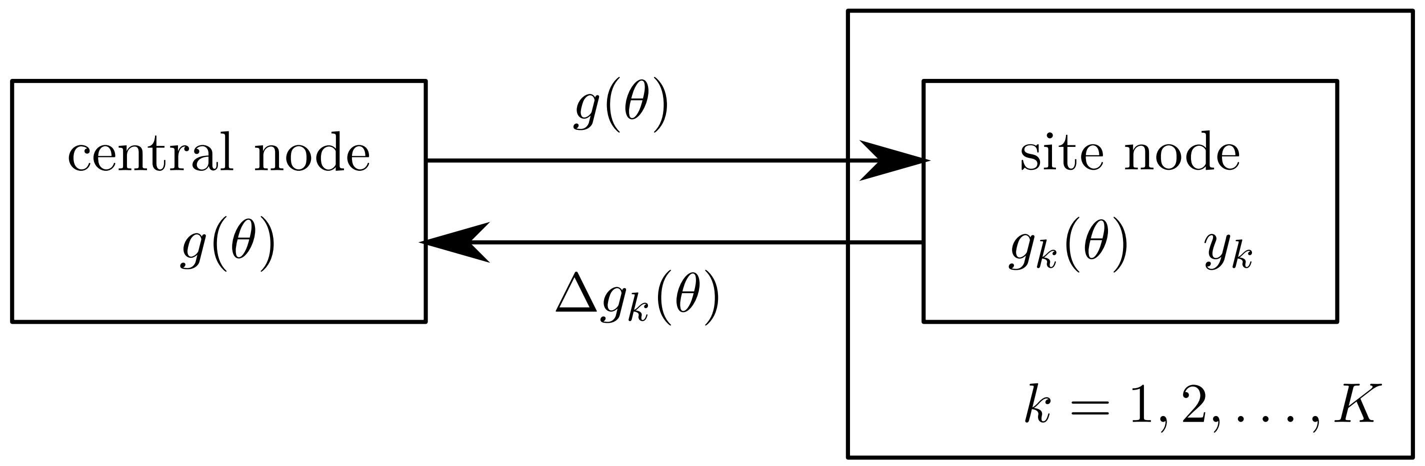


Figure 1: The EP framework for partitioned data. The central node stores the current parameters for the global approximation $g(\theta)$. Each site node $k = 1, 2, \ldots, K$ stores the current parameters for the site approximation $g_k(\theta)$ and the assigned partition of the data y_k. The central node sends the parameters of $g(\theta)$ to the site nodes. In parallel, the site nodes update $g_k(\theta)$ and send back the difference in the parameters.

The current g is needed at all sites

Makes for a simple distributed architecture

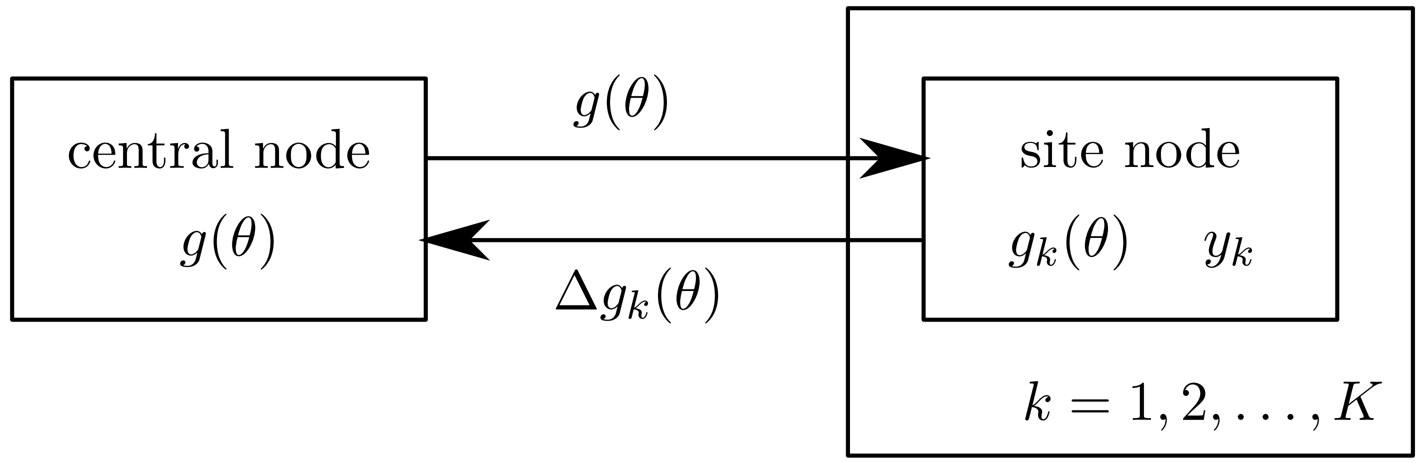


Figure 1: The EP framework for partitioned data. The central node stores the current parameters for the global approximation $g(\theta)$. Each site node $k = 1, 2, \ldots, K$ stores the current parameters for the site approximation $g_k(\theta)$ and the assigned partition of the data y_k. The central node sends the parameters of $g(\theta)$ to the site nodes. In parallel, the site nodes update $g_k(\theta)$ and send back the difference in the parameters.

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Scaling Distributed Machine Learning with the Parameter Server

Mu Li, *Carnegie Mellon University and Baidu;* David G. Andersen and Jun Woo Park, *Carnegie Mellon University;* Alexander J. Smola, *Carnegie Mellon University and Google, Inc.;* Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su, *Google, Inc.*

https://www.usenix.org/conference/osdi14/technical-sessions/presentation/li_mu

This paper is included in the Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation.

October 6–8, 2014 • Broomfield, CO

978-1-931971-16-4

The current g is needed at all sites

Makes for a simple distributed architecture

Vehtari et al.

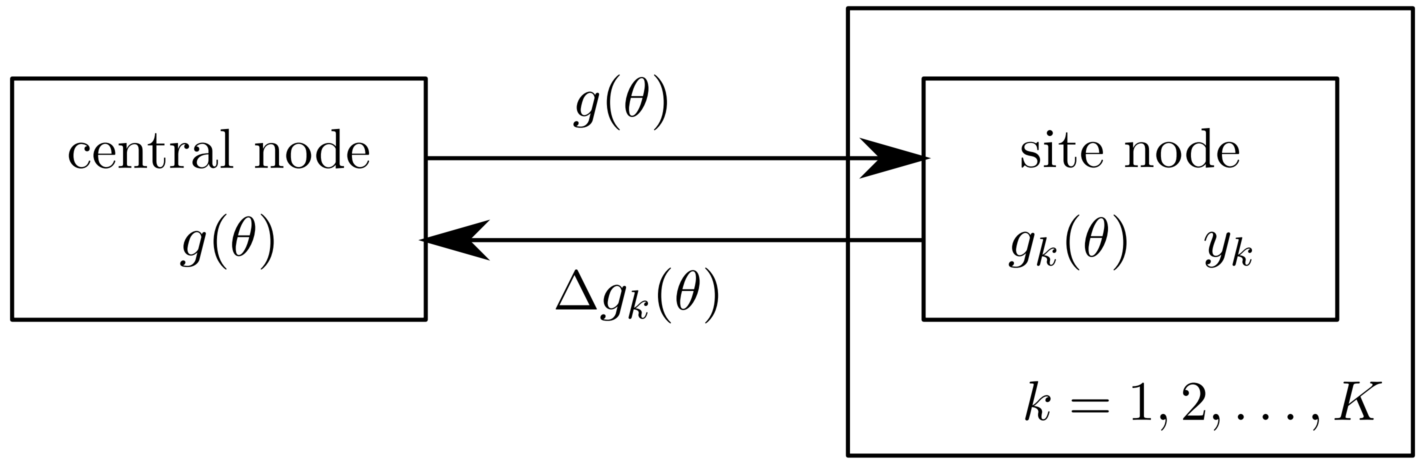


Figure 1: The EP framework for partitioned data. The central node stores the current parameters for the global approximation $g(\theta)$. Each site node $k = 1, 2, \ldots, K$ stores the current parameters for the site approximation $g_k(\theta)$ and the assigned partition of the data y_k. The central node sends the parameters of $g(\theta)$ to the site nodes. In parallel, the site nodes update $g_k(\theta)$ and send back the difference in the parameters.

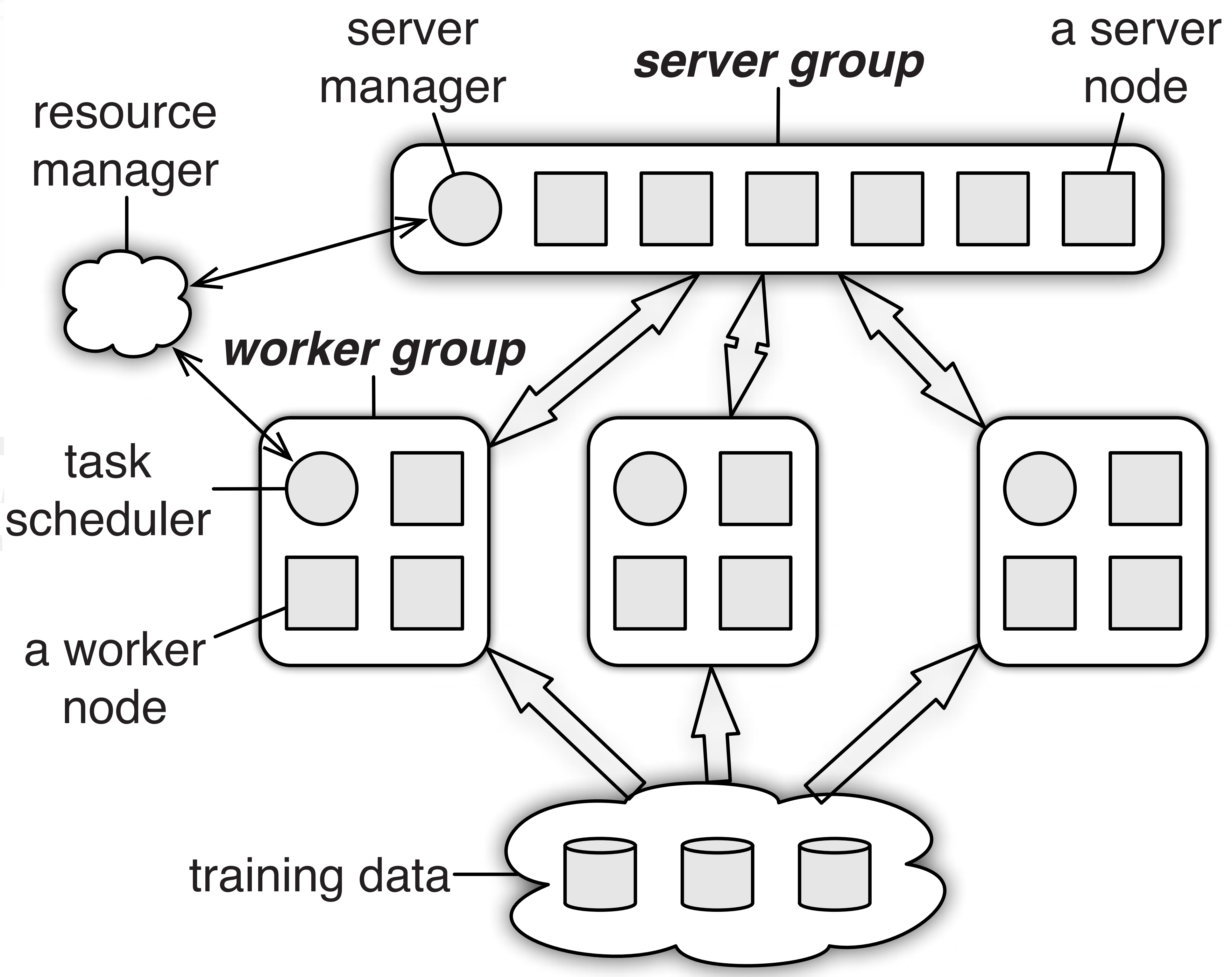


Figure 4: Architecture of a parameter server communicating with several groups of workers.

Tradeoffs and considerations

- Data partitioning: More sites = more parallelism, but worse approximations
- Exact form of g: need not be Gaussian, often is
- Initial estimates influence convergence
- How to estimate $g_{/k}$ (Vehtari &al. do MCMC, the original EP was closed-form)
- Asynchronous updates would be nice if some sites are small
- Damping of updates to global g? (analogous to step size in gradient descent)
- Potential numerical stability issues working with covariance matrices

Tradeoffs and considerations

No free lunches!