Carnegie Mellon

School of Computer Science

Cut-And-Stitch: Efficient Parallel Learning of Linear Dynamical Systems on SMPs

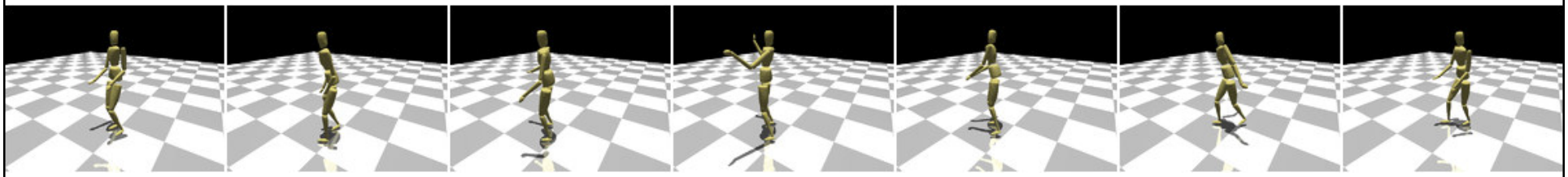
Lei Li, Wenjie Fu, Fan Guo,
Todd C. Mowry, Christos Faloutsos
Computer Science Department
Carnegie Mellon University
leili@cs.cmu.edu

Carnegie Mellon

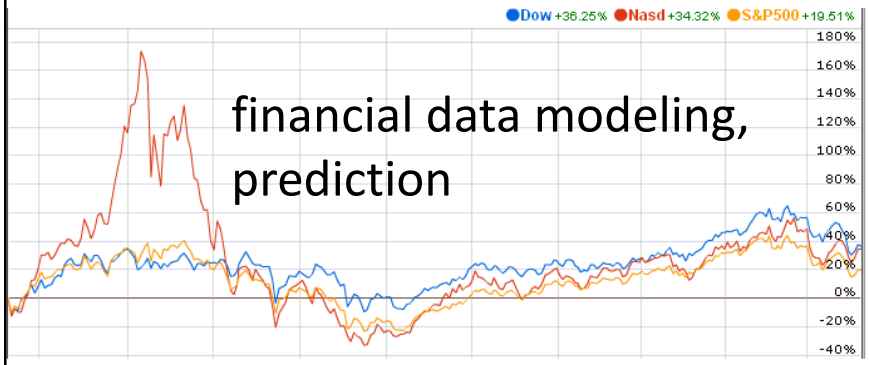
Outline

- Motivation
- Background
- Proposed Method: Cut-And-Stitch
- Experiments
- Conclusion

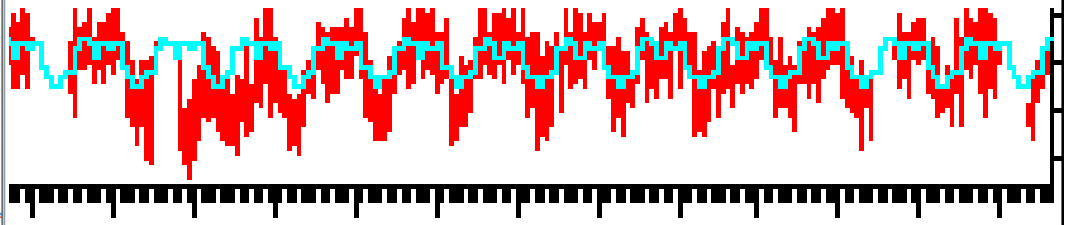
Motivation



Motion Capture, Human Motion Modeling



financial data modeling, prediction



sensor data, pattern mining

- Common tool: Linear Dynamical System (LDS)
- Challenge?
  - Learning LDS is slow

Need For Speed

Carnegie Mellon

Related Work

- Parallel mining of closed sequential patterns, [KDD 05, S. Cong, J. Han, D. Padua]
- Mining terabytes of data for frequent itemsets, [PPoPP 07, G. Buehrer, S. Parthasarathy, S. Tatikonda, T. Kurc, J. Saltz]
- Parallelize the discovery of frequent patterns in large graphs, [IPDPS 07, S. Reinhardt, G. Karypis]
- Parallel algorithms for SVM
  - Cascade SVM, [NIPS 04, H. Graf, E. Cosatto, L. Bottou, I. Durdanovic, V. Vapnik]
  - Parallel Mixture of SVMs, [NIPS 02, R. Collobert, S. Bengio, Y. Bengio]
  - PSVM (open src), [NIPS 07, E. Chang, K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, H. Cui]
- Multicore Map-Reduce approach, [NIPS 06, C. Chu, S. Kim, Y. Lin, Y. Yu, G. Bradski, A. Ng, K. Olukotun]

Carnegie Mellon

Problem Definition

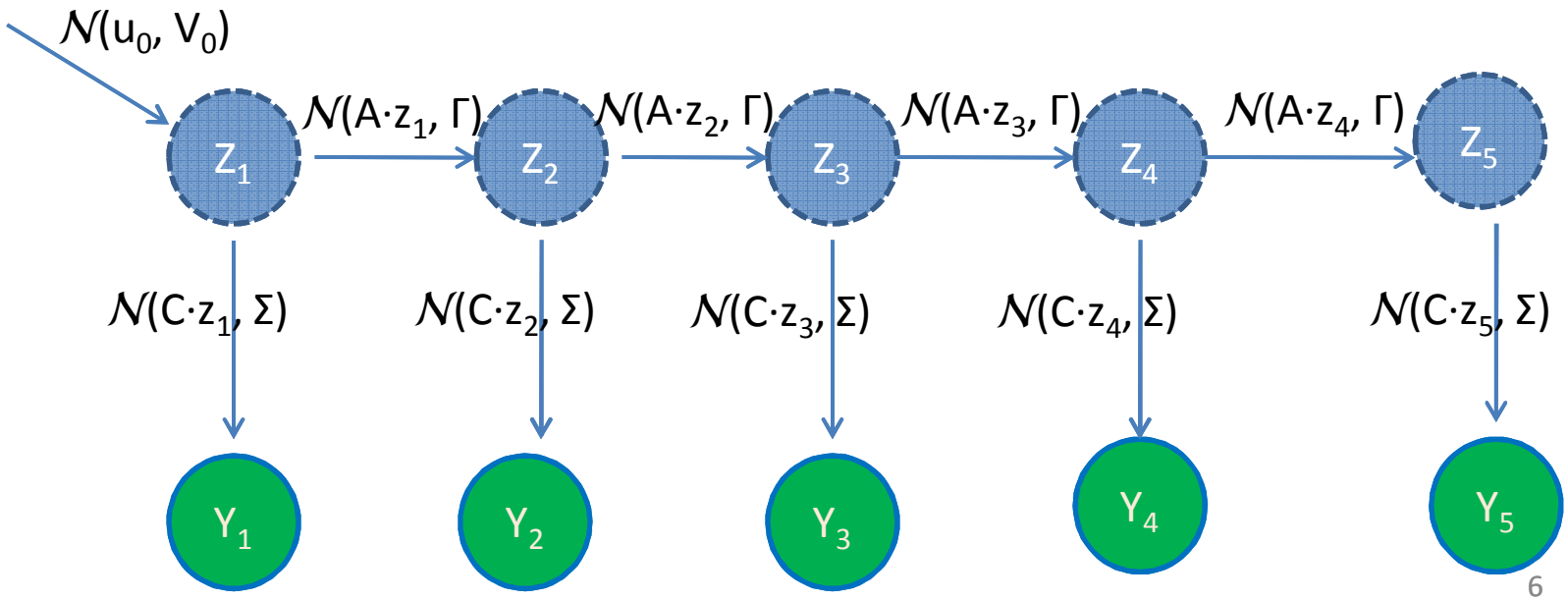
- Problem:
  - **Given** a sequence of data, **find** the best model parameters for Linear Dynamical System
- Traditional Method:
  - *Maximum Likelihood* Estimation via *Expectation-Maximization*(EM) algorithm
- Objective:
  - **Parallelize** the learning algorithm
- Assumption:
  - *shared memory architecture*

Carnegie Mellon

Linear Dynamical System

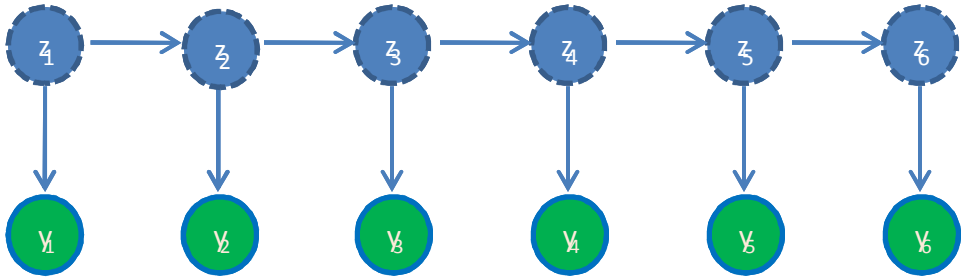
aka. Kalman Filter

- Parameters: $\theta=(u_0, V_0, A, \Gamma, C, \Sigma)$
- Observation: $y_1 \ldots y_n$
- Hidden variables: $z_1 \ldots z_n$

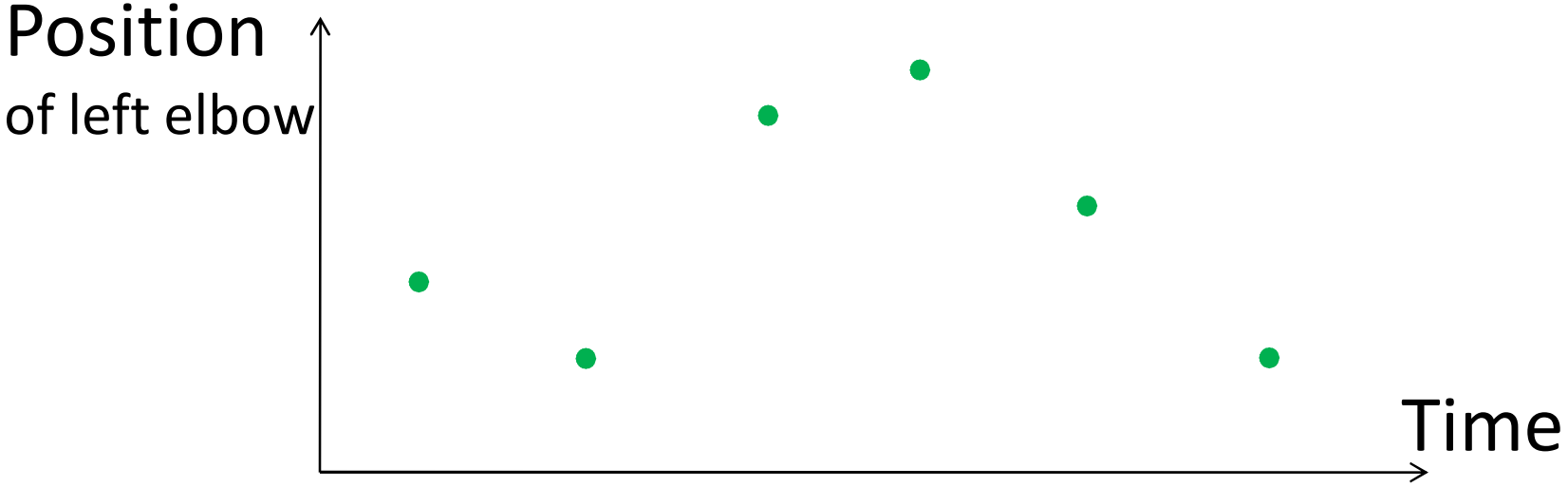


Carnegie Mellon

Example



given positions, estimate dynamics (i.e. params)

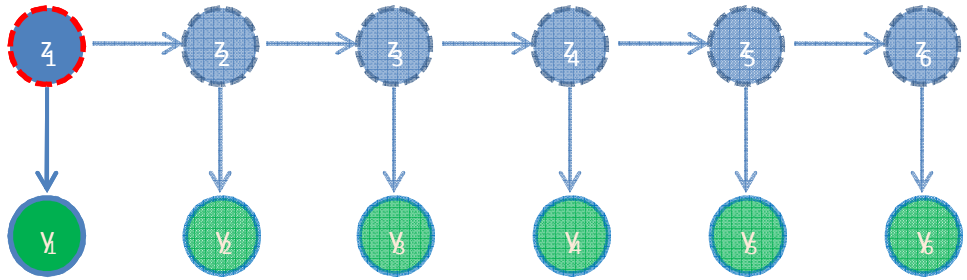


Carnegie Mellon

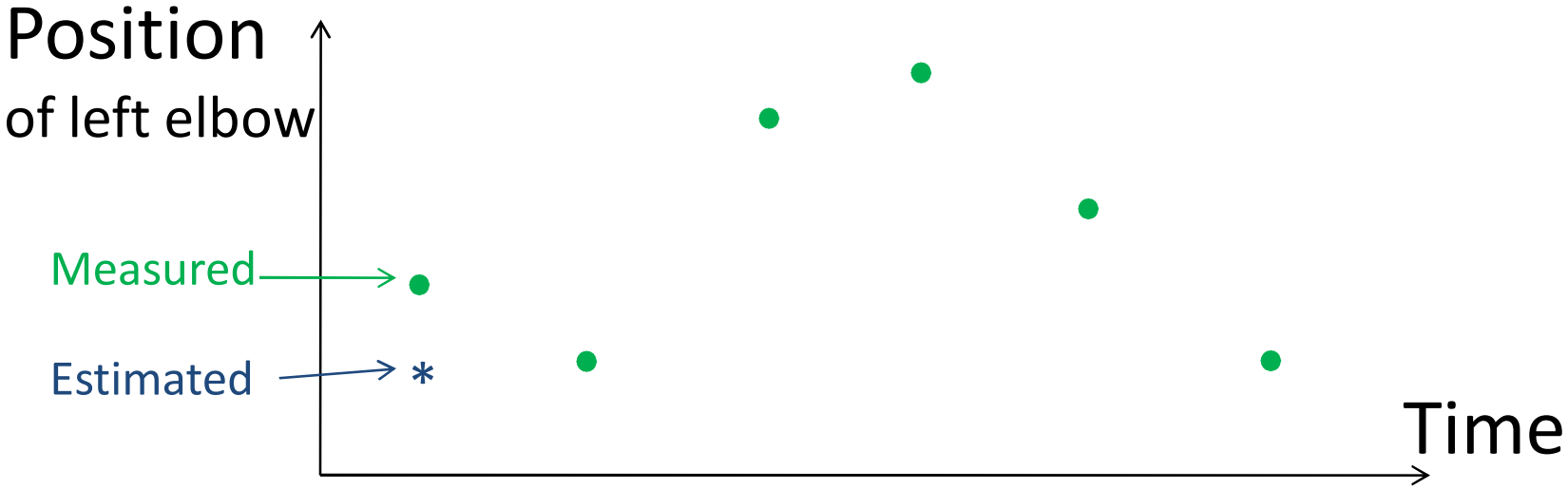
Traditional:
How to learn LDS?

Carnegie Mellon

Sequential Learning (EM)

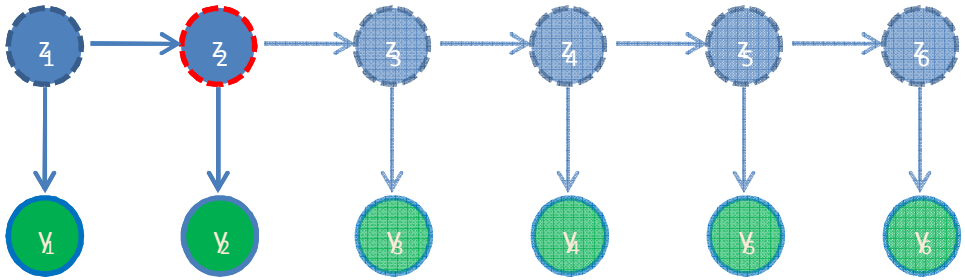


Compute $P(z_1 \mid y_1)$

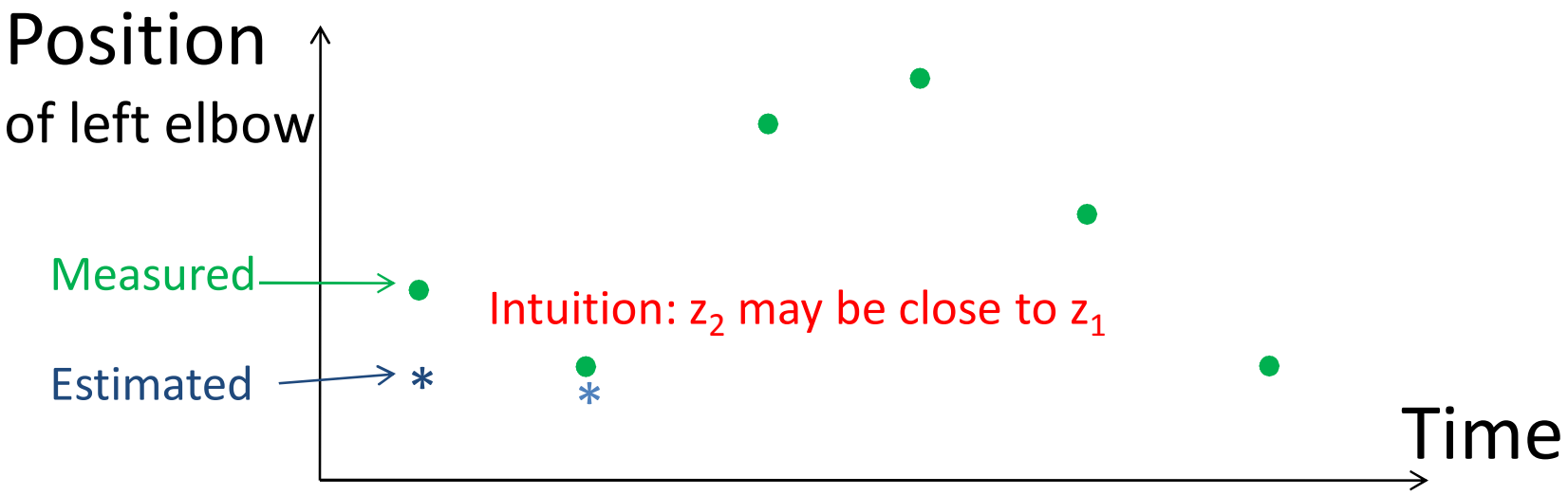


Carnegie Mellon

Sequential Learning (EM)

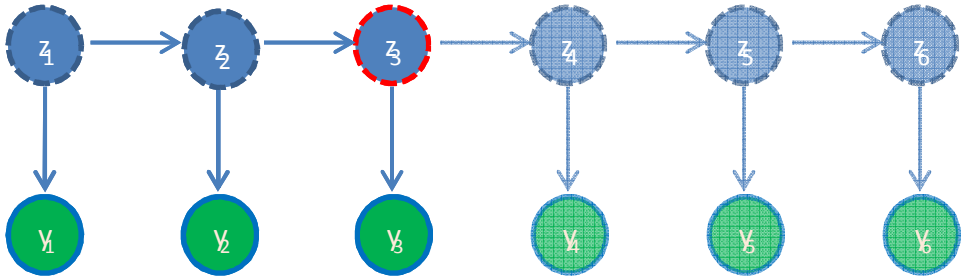


From $P(z_1 \mid y_1)$ → Compute $P(z_2 \mid y_1, y_2)$

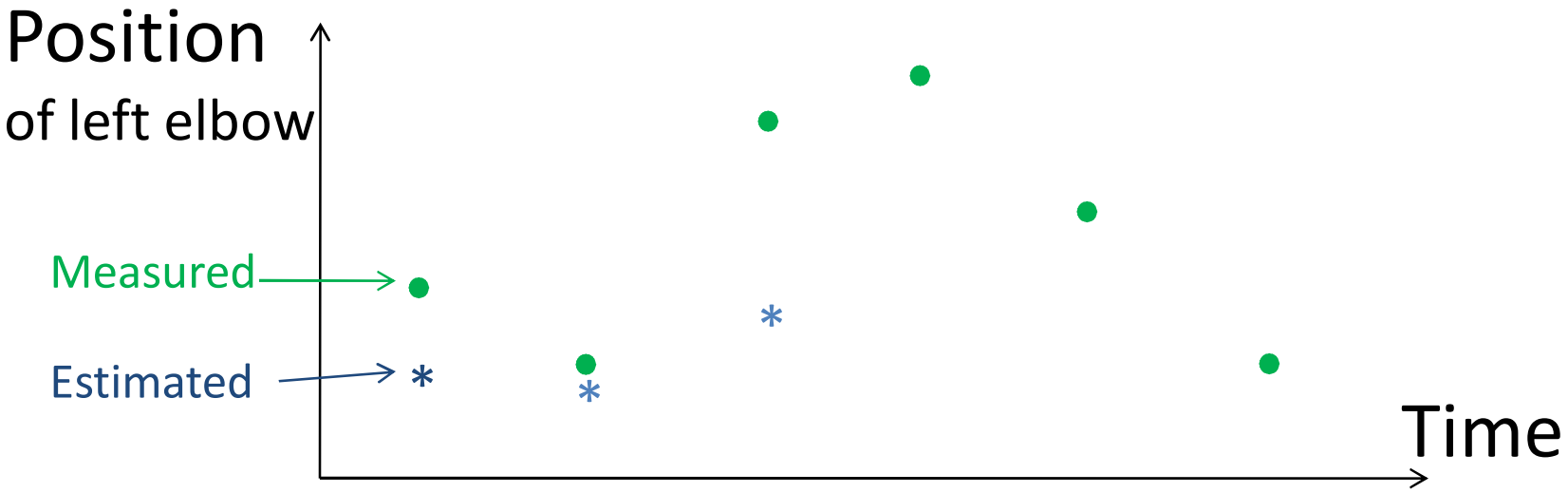


Carnegie Mellon

Sequential Learning (EM)

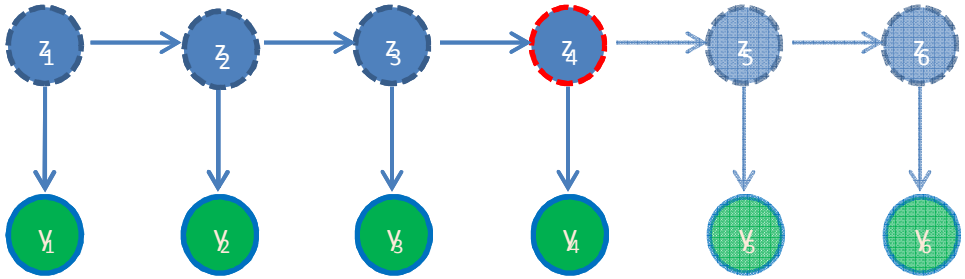


From $P(z_2 | y_1, y_2)$ → Compute $P(z_3 | y_1, y_2, y_3)$

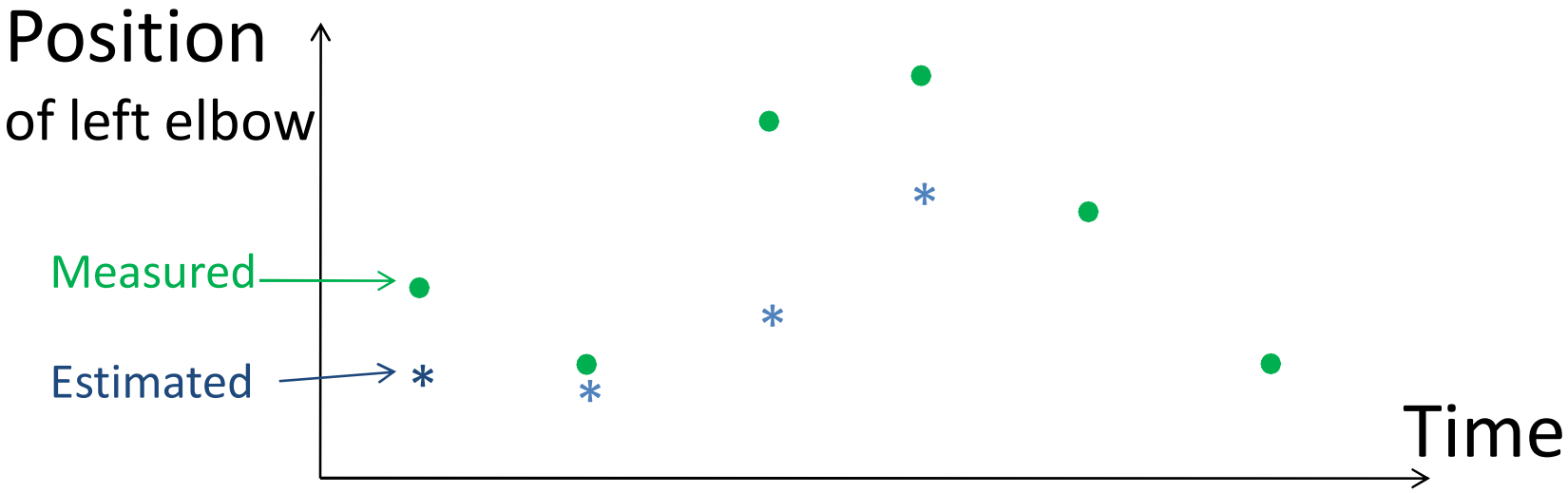


Carnegie Mellon

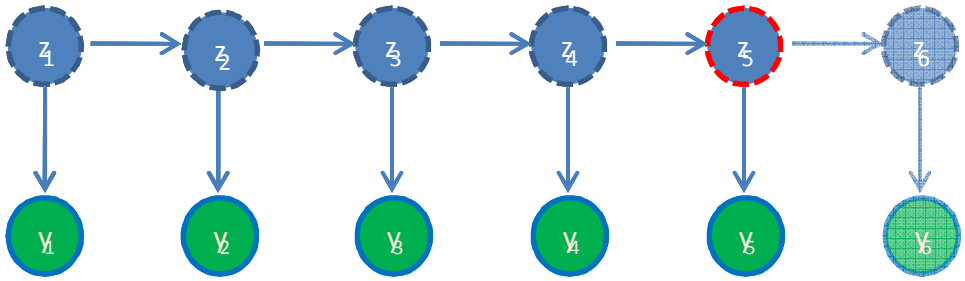
Sequential Learning (EM)



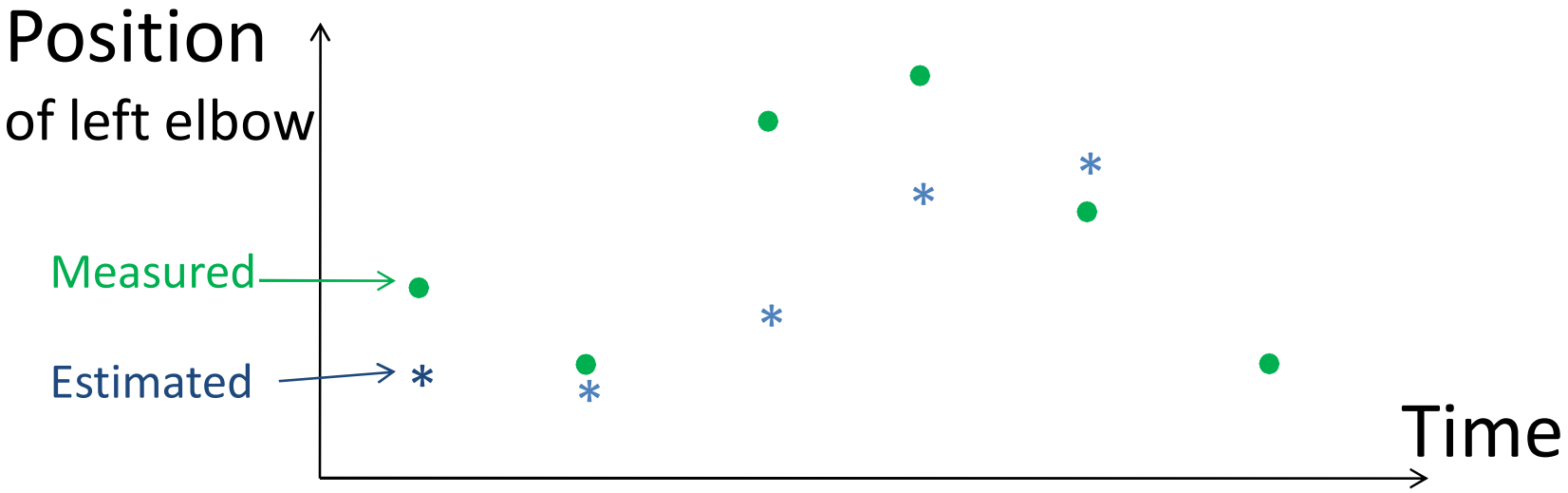
From $P(z_3 | y_1, y_2, y_3)$ → Compute $P(z_4 | y_1, y_2, y_3, y_4)$



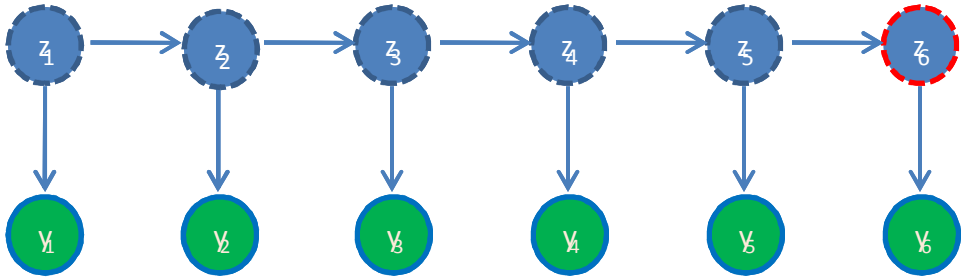
Sequential Learning (EM)



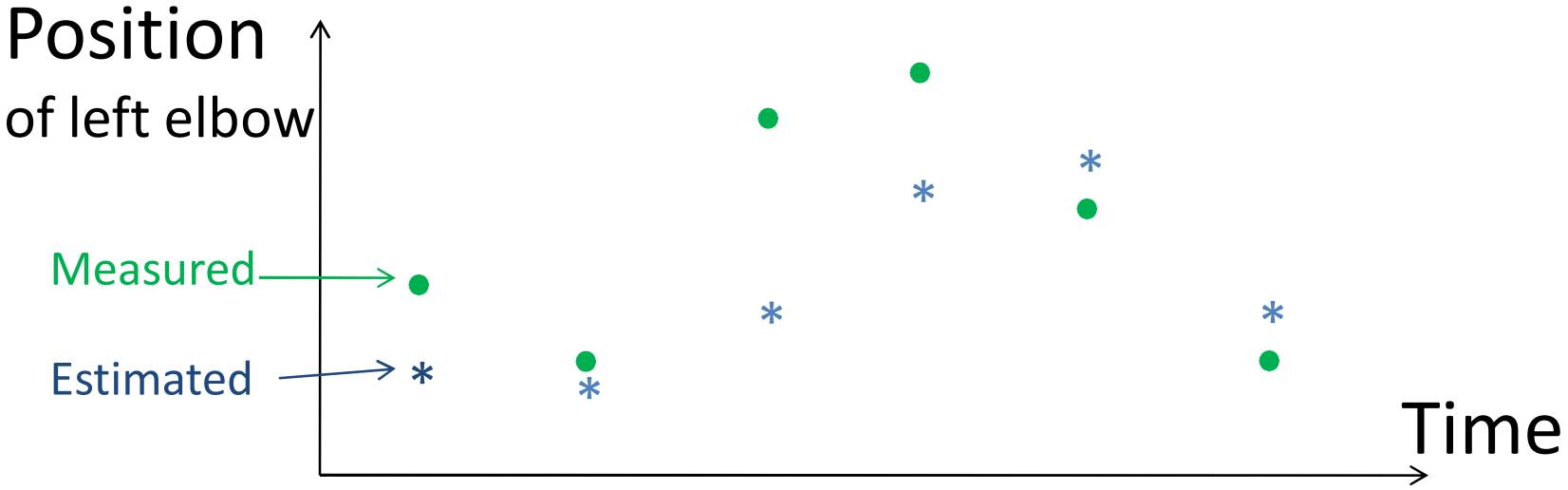
From $P(z_4 | y_1, y_2, y_3, y_4)$ → Compute $P(z_5 | y_1, y_2, y_3, y_4, y_5)$



Sequential Learning (EM)

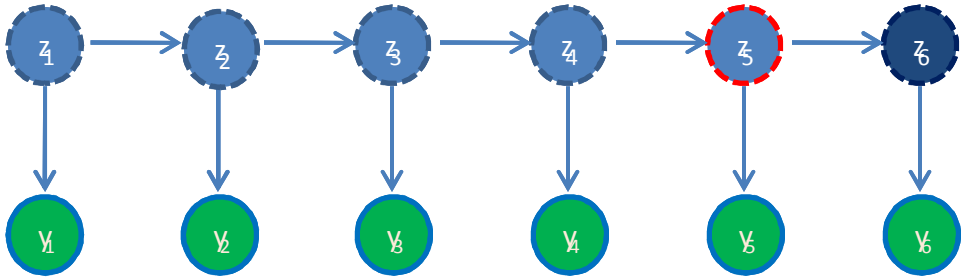


From $P(z_5 | y_1, y_2, y_3, y_4, y_5)$ → Compute $P(z_6 | y_1, y_2, y_3, y_4, y_5, y_6)$

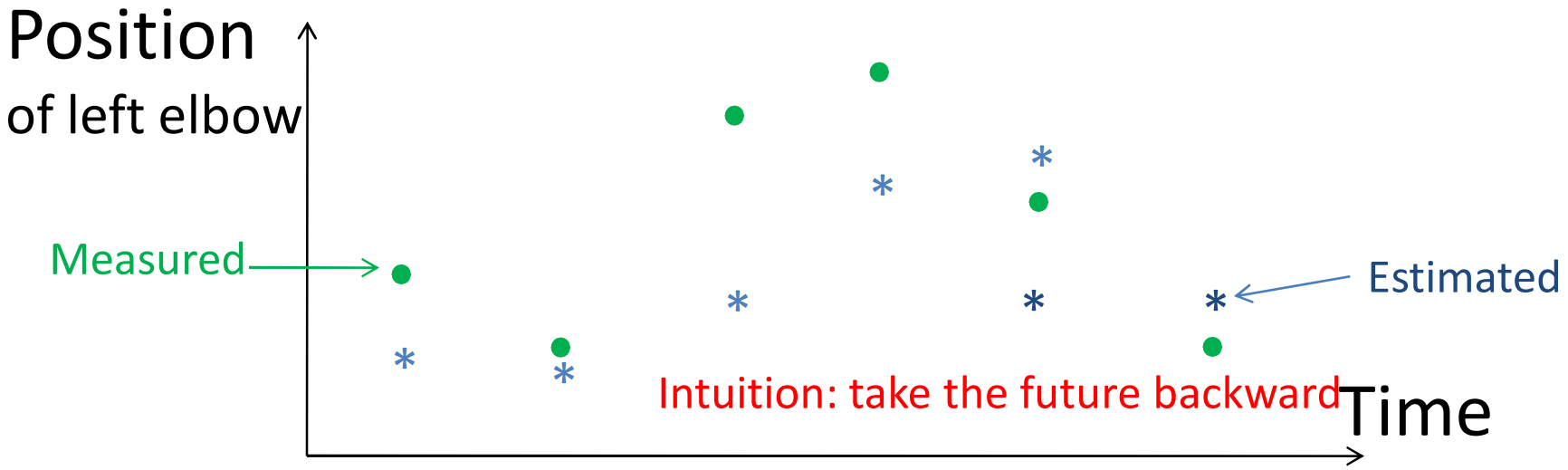


Carnegie Mellon

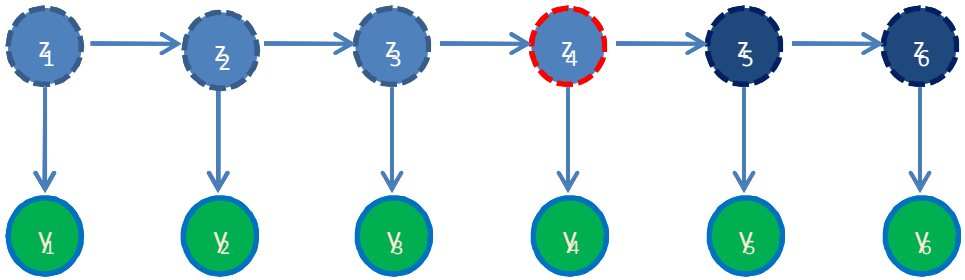
Sequential Learning (EM)



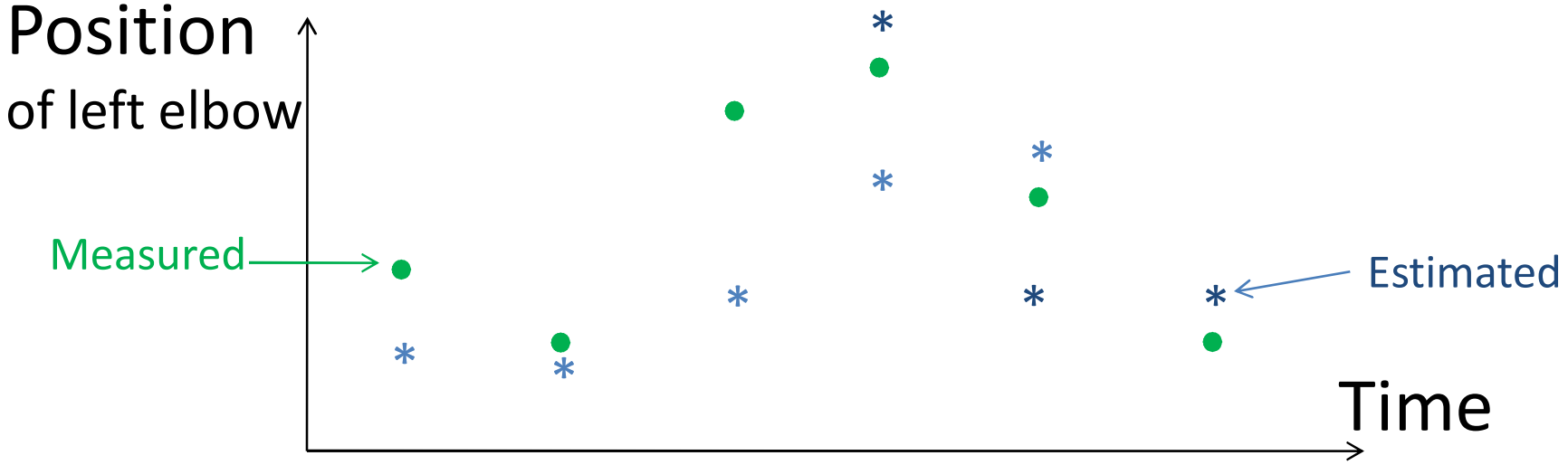
From $P(z_6 | y_1, y_2, y_3, y_4, y_5, y_6)$ → Compute $P(z_5 | y_1, y_2, y_3, y_4, y_5, y_6)$



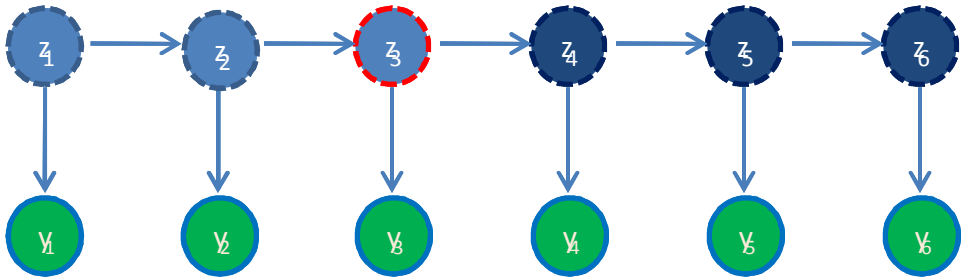
Sequential Learning (EM)



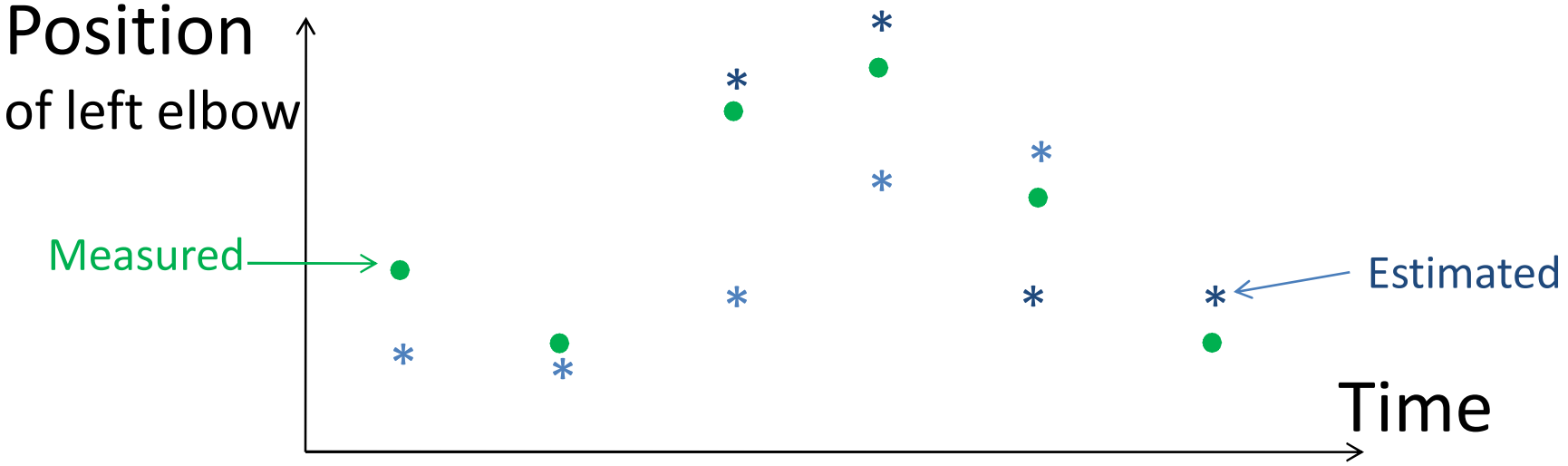
From $P(z_6 | y_1, y_2, y_3, y_4, y_5, y_6)$ → Compute $P(z_4 | y_1, y_2, y_3, y_4, y_5, y_6)$



Sequential Learning (EM)

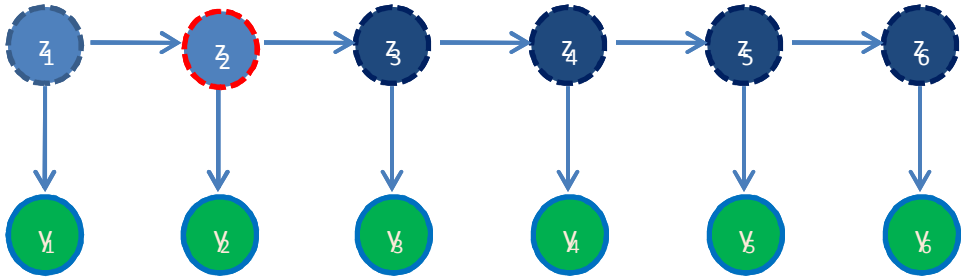


From $P(z_4 | y_1, y_2, y_3, y_4, y_5, y_6)$ → Compute $P(z_3 | y_1, y_2, y_3, y_4, y_5, y_6)$

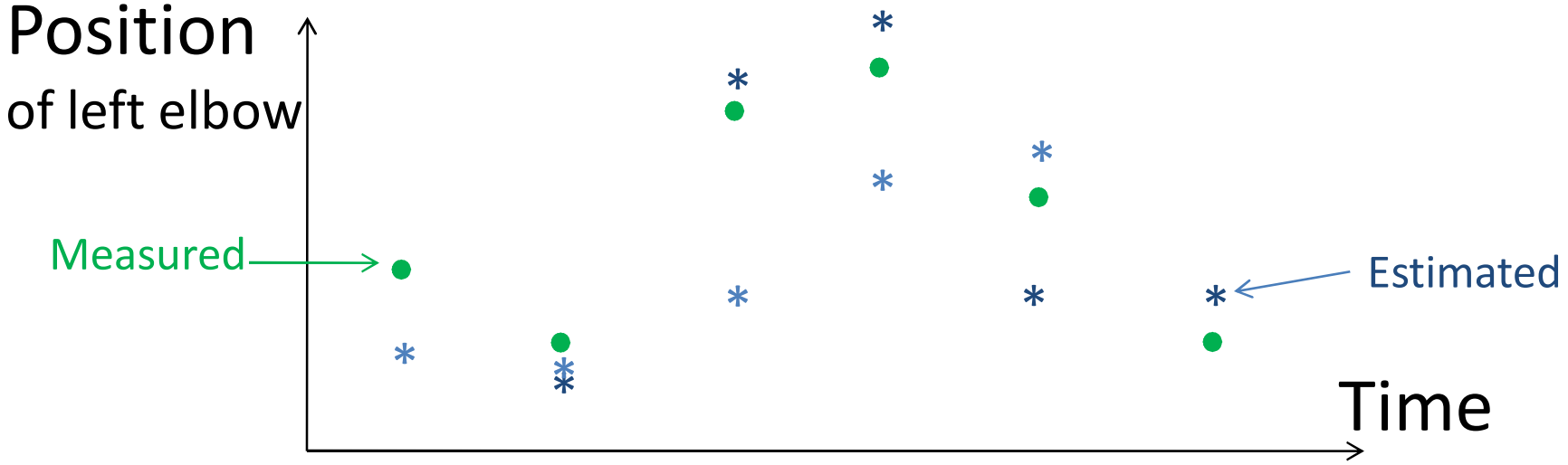


Carnegie Mellon

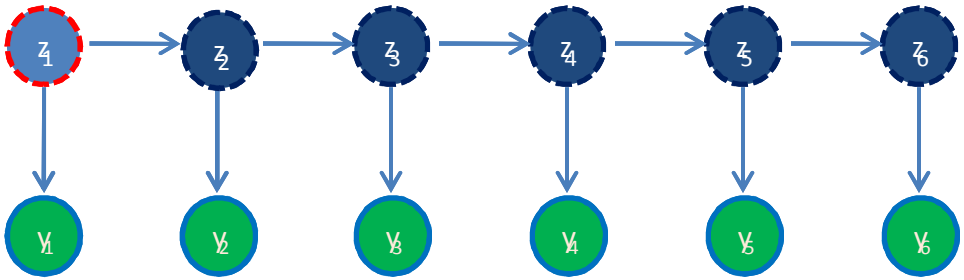
Sequential Learning (EM)



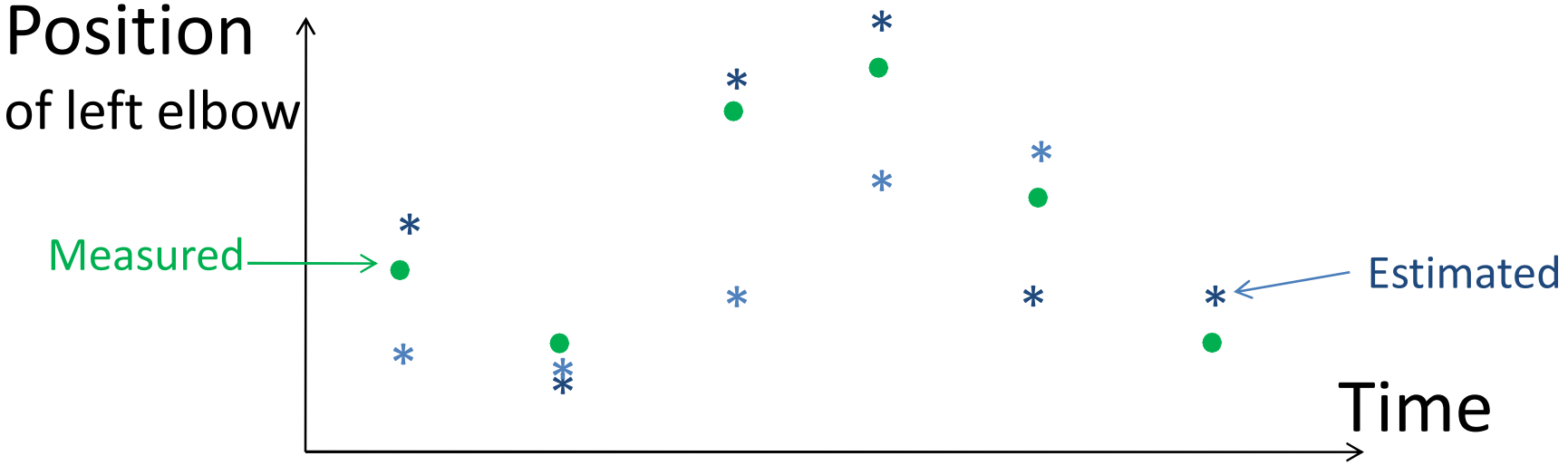
From $P(z_3 | y_1, y_2, y_3, y_4, y_5, y_6)$ → Compute $P(z_2 | y_1, y_2, y_3, y_4, y_5, y_6)$



Sequential Learning (EM)

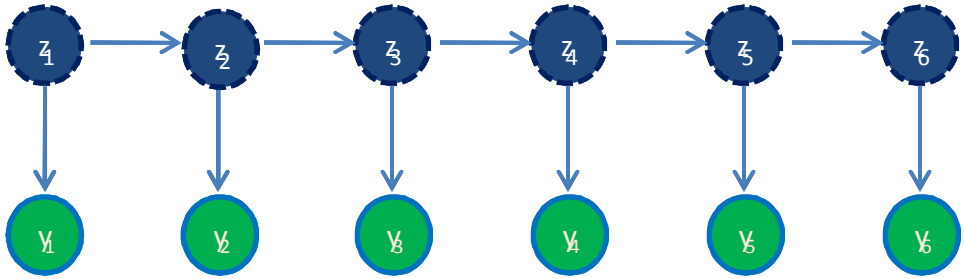


From $P(z_2 | y_1, y_2, y_3, y_4, y_5, y_6)$ → Compute $P(z_1 | y_1, y_2, y_3, y_4, y_5, y_6)$



Carnegie Mellon

Sequential Learning (EM)



From all posterior $z_1, z_2, z_3, z_4, z_5, z_6$

$P(z_1 | y_1, y_2, y_3, y_4, y_5, y_6)$, $P(z_2 | y_1, y_2, y_3, y_4, y_5, y_6)$…

Compute sufficient statistics

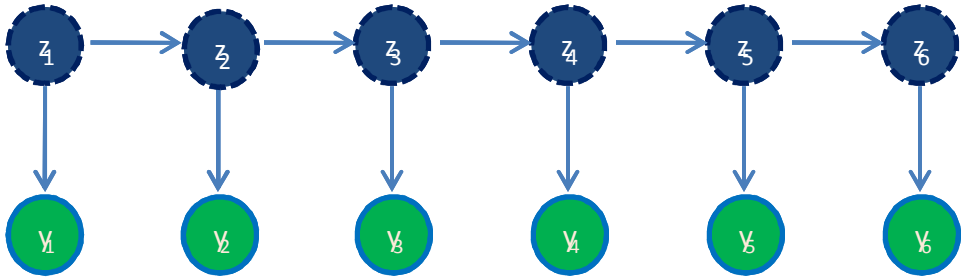
$E[z_i]$

$E[z_i z_i']$

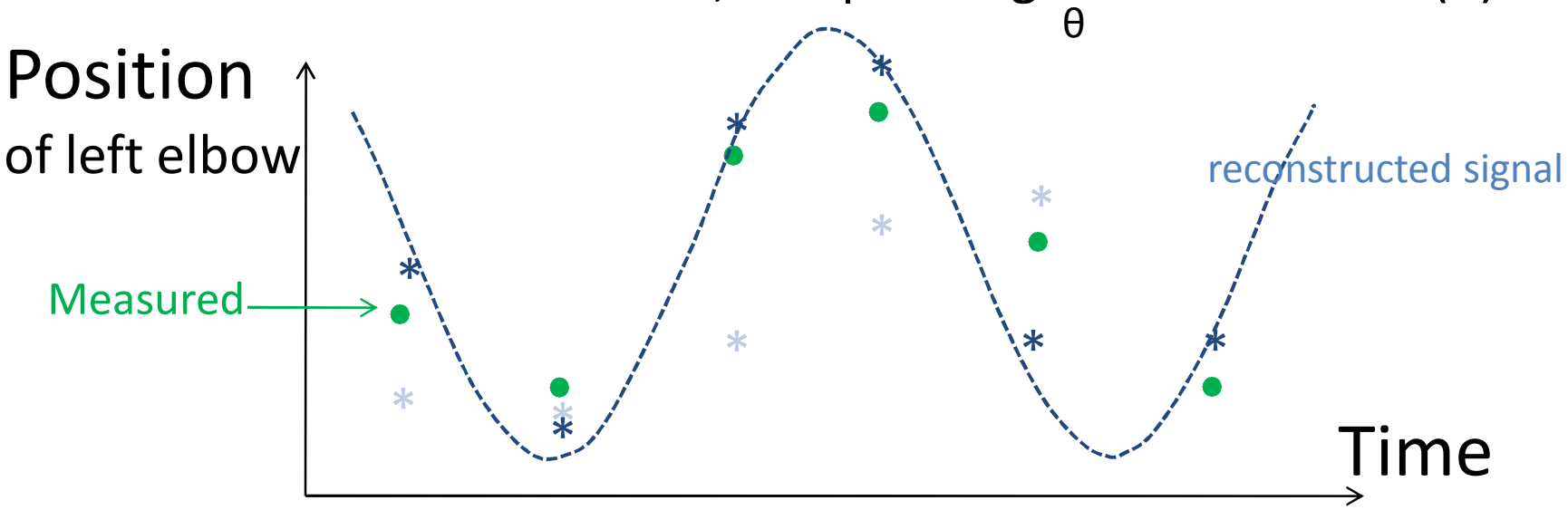
$E[z_{i-1} z_i']$

Carnegie Mellon

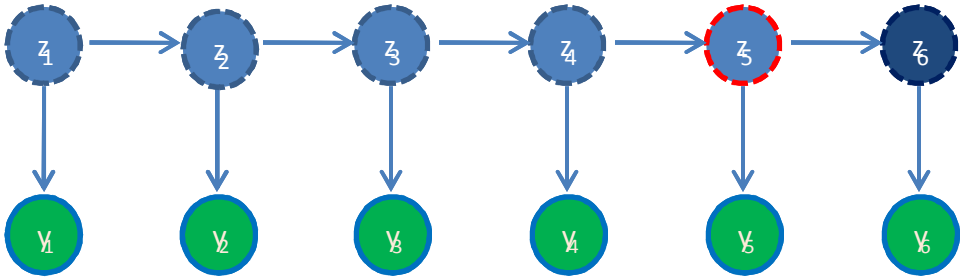
Sequential Learning (EM)



with sufficient statistics, compute argmax ←likelihood(θ)

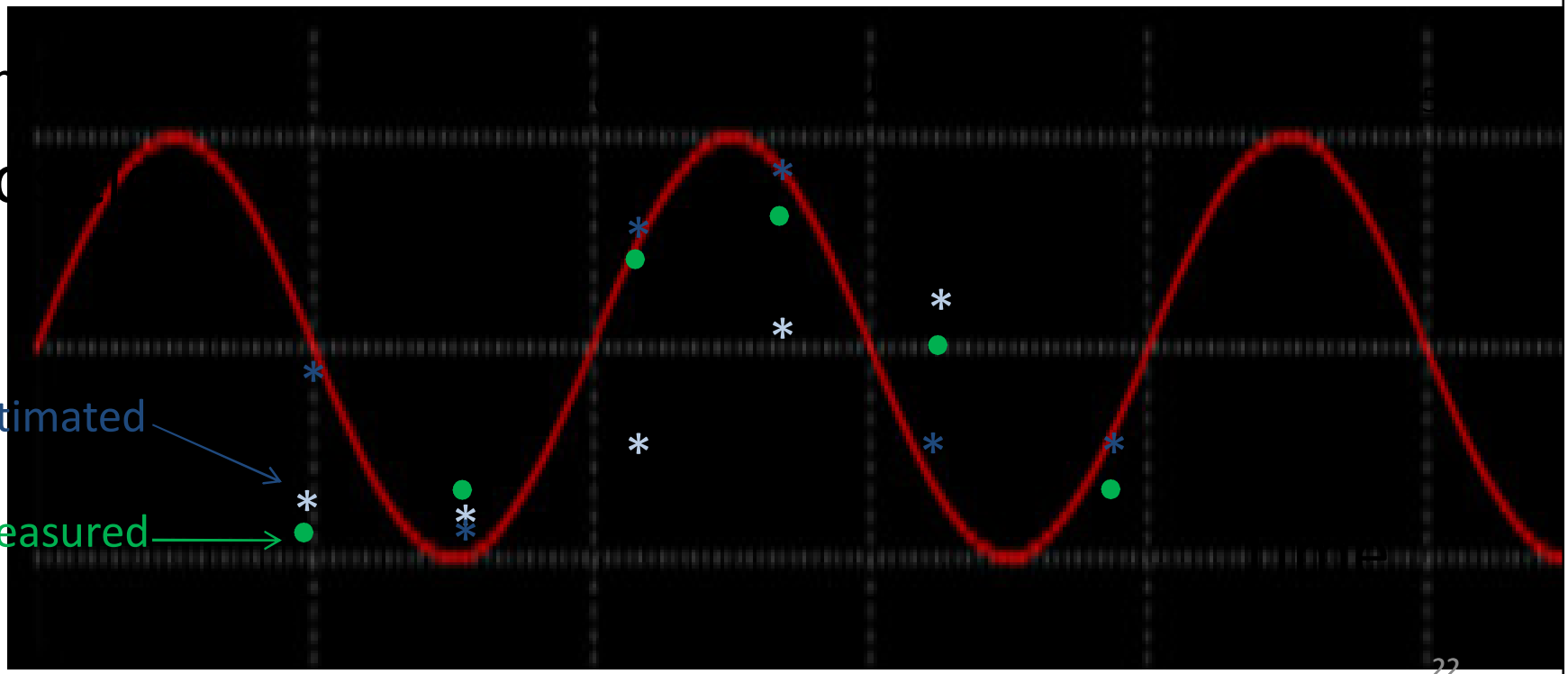


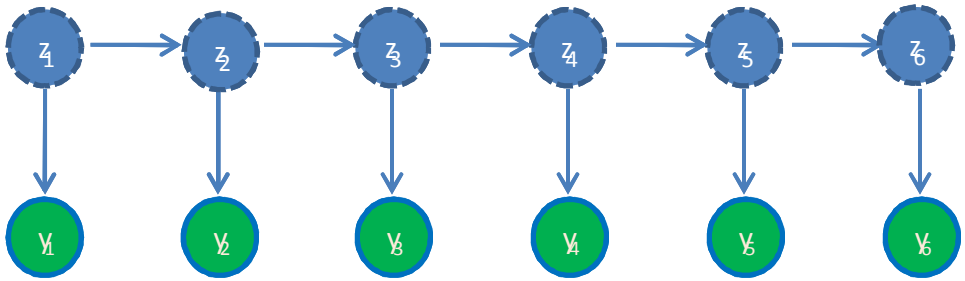
Sequential Learning (EM)



From

Pc





Speed Bottleneck:

sequential computation of posterior

How to *parallelize* it?

Carnegie Mellon

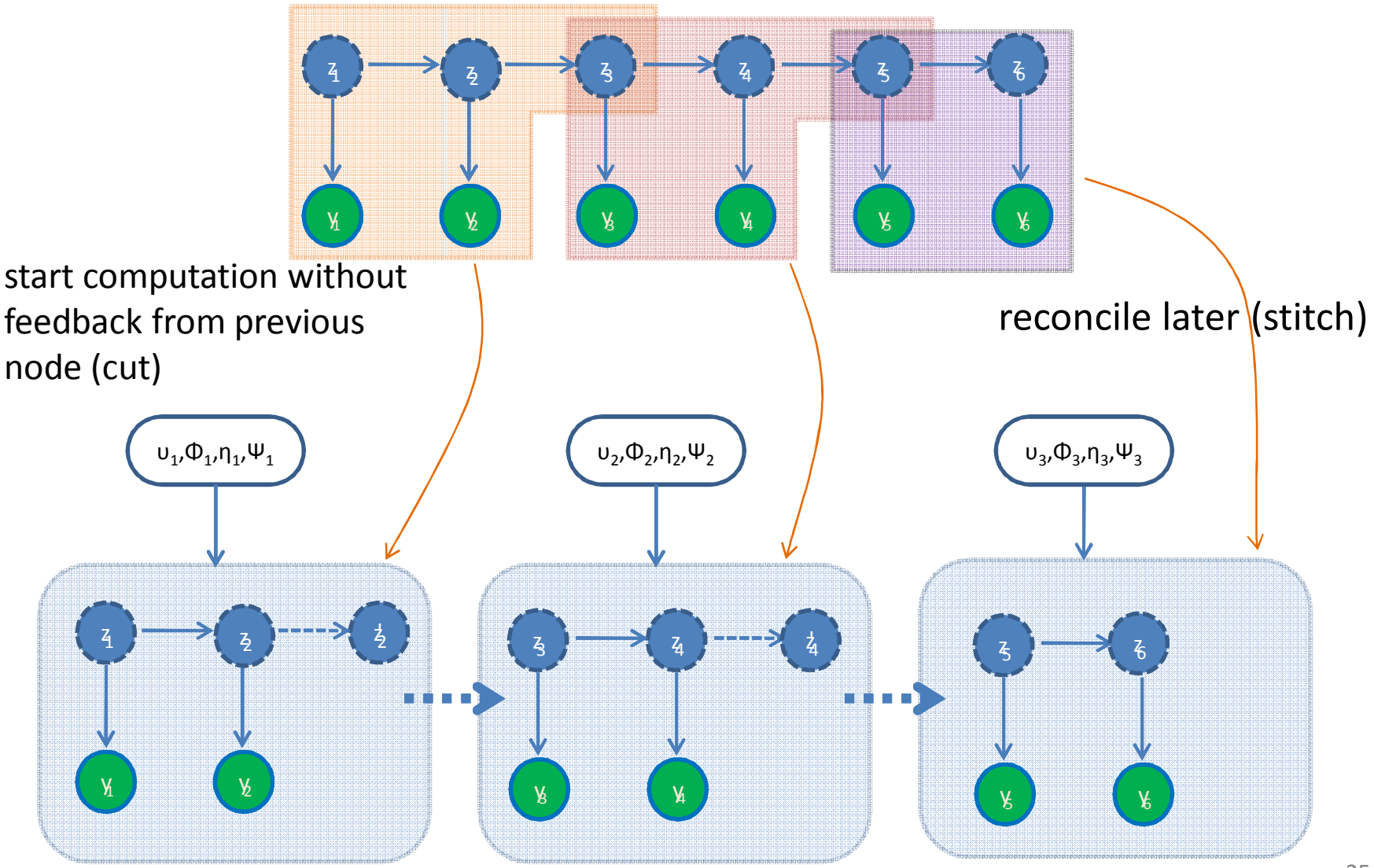
"Leap of faith"

start computation without feedback from previous node (cut),

and reconcile later (stitch)

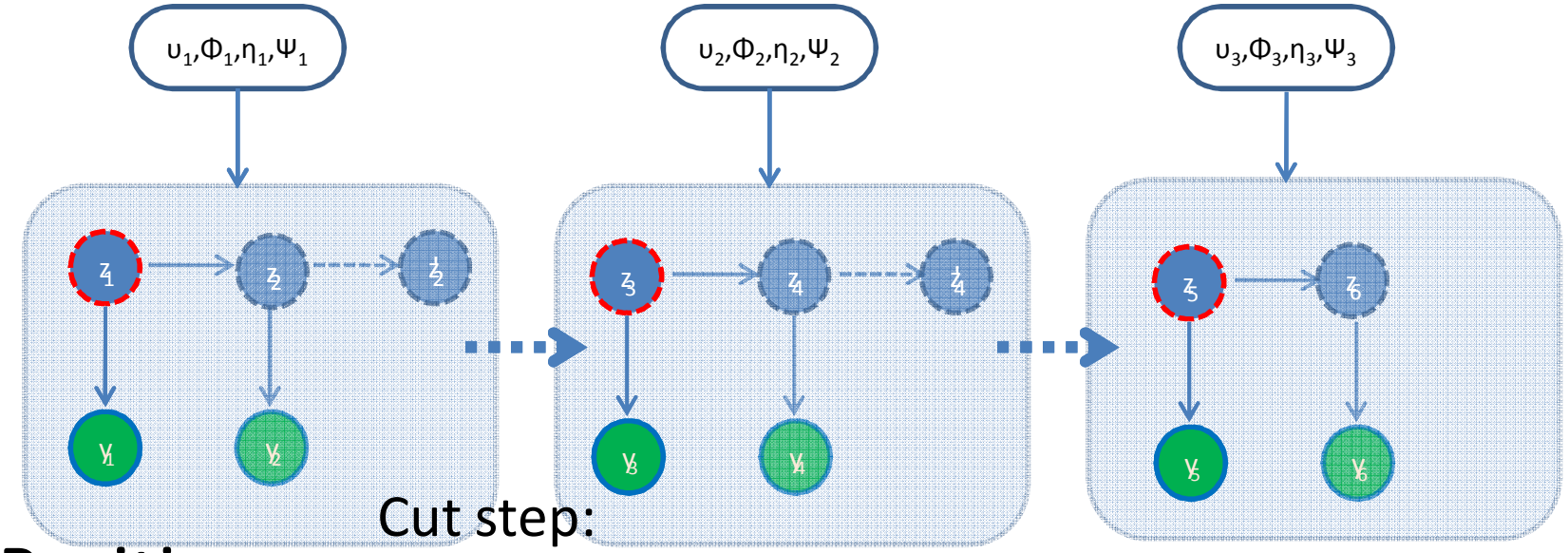
Carnegie Mellon

Proposed Method: Cut-And-Stitch



Carnegie Mellon

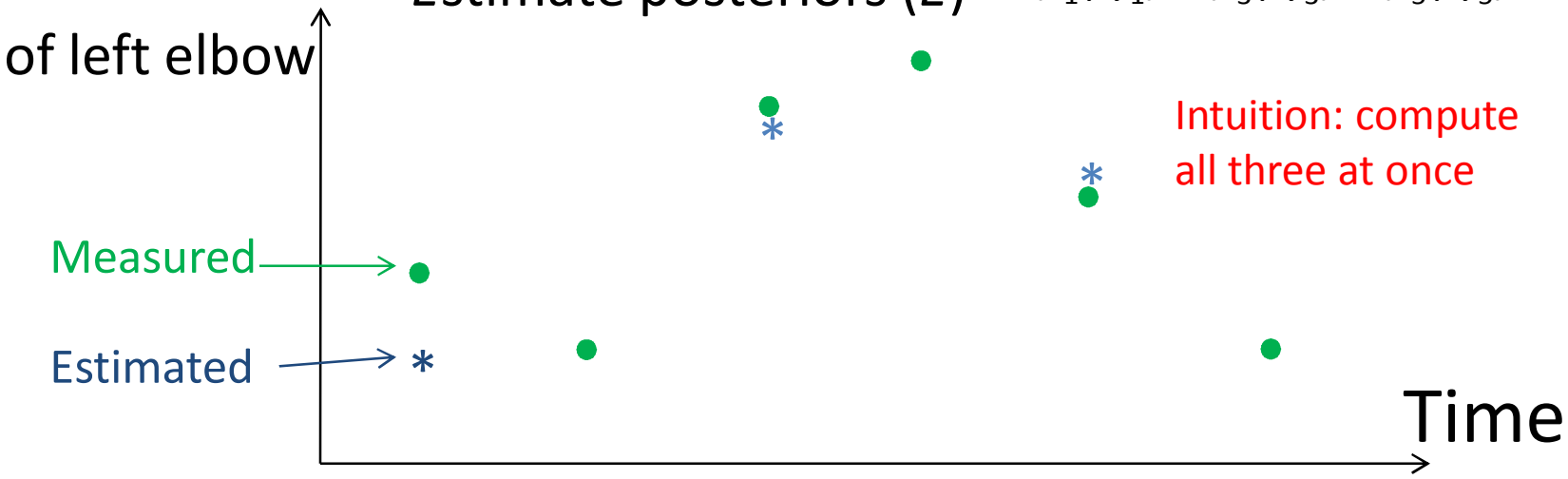
Cut-And-Stitch



Cut step:

*E*stimate posteriors (*E*) $P(z_1 | y_1), P(z_3 | y_3), P(z_5 | y_5)$

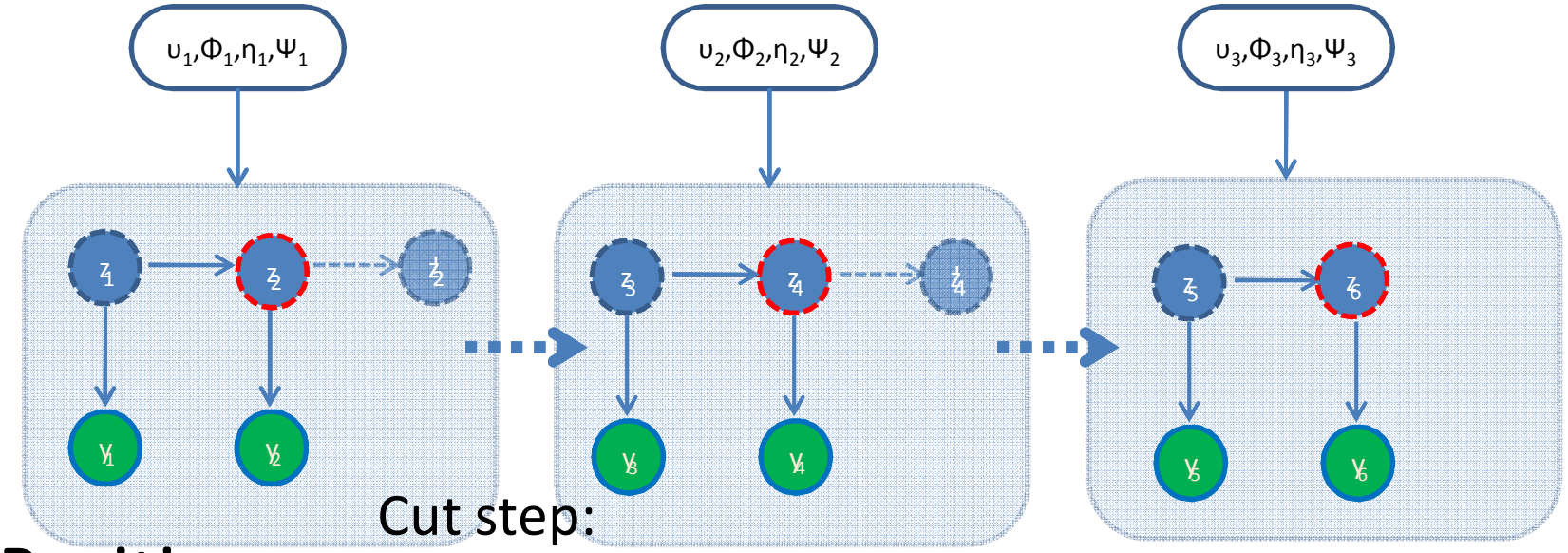
Position of left elbow



Intuition: compute all three at once

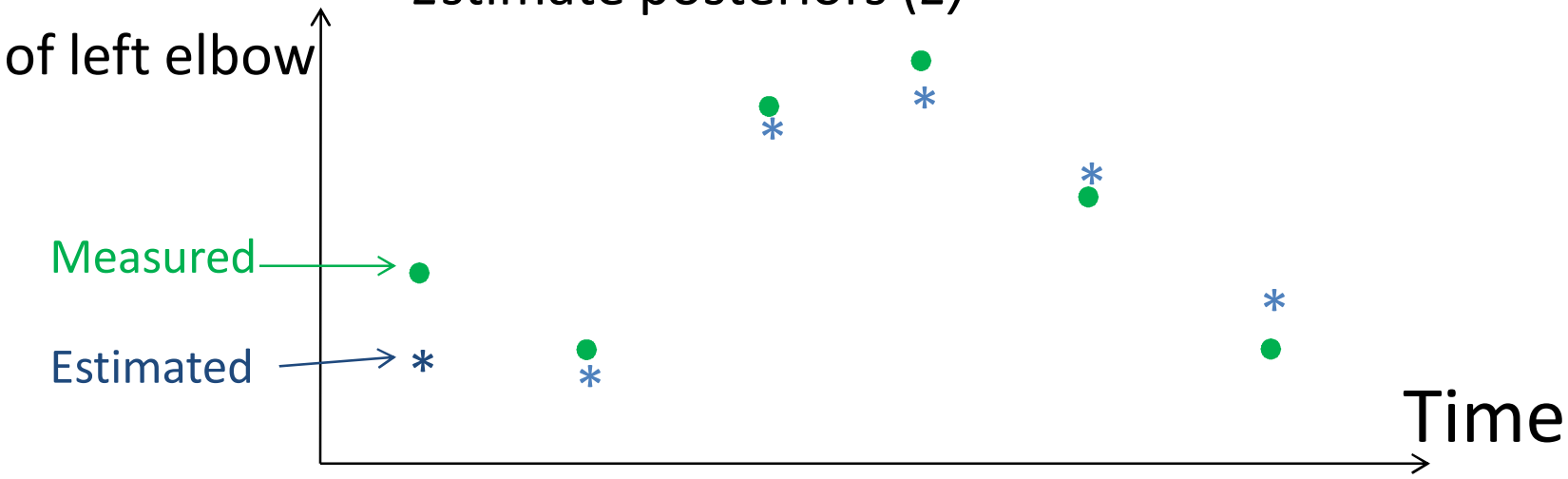
Carnegie Mellon

Cut-And-Stitch



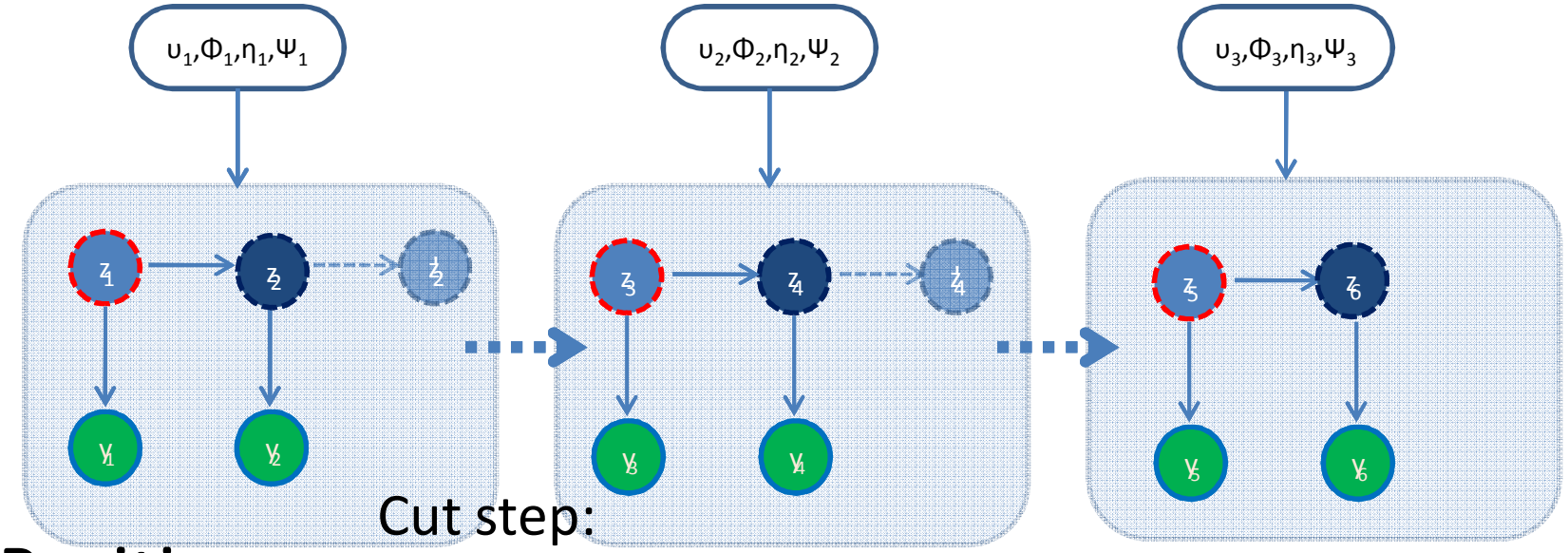
Cut step:

*E*stimate posteriors (*E*)



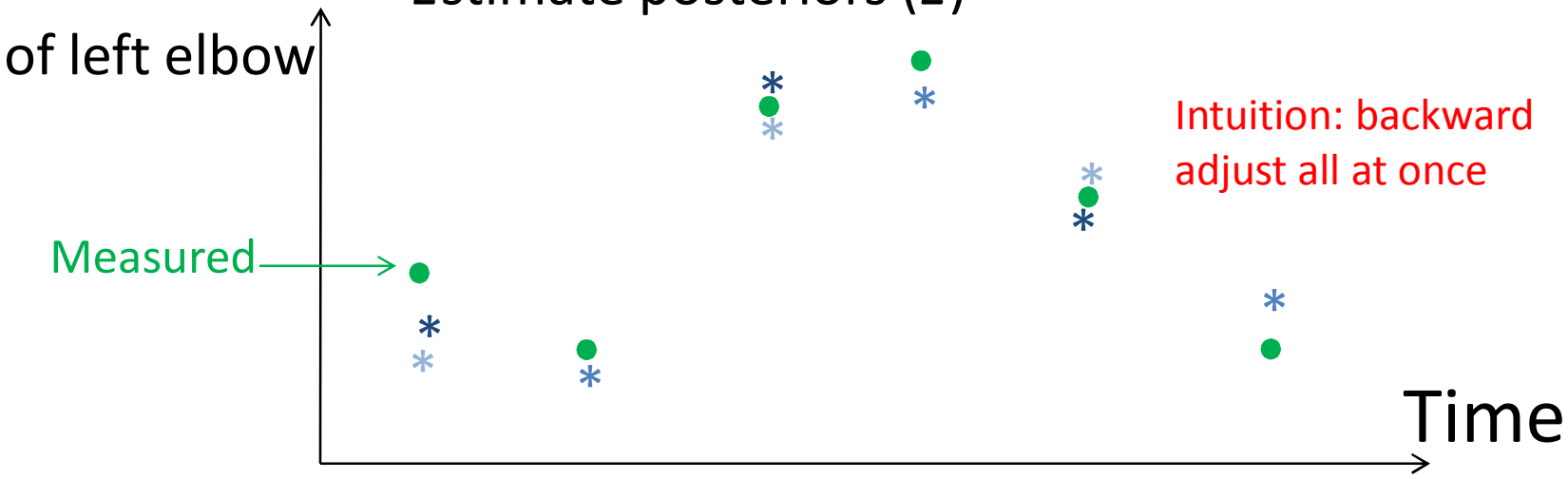
Carnegie Mellon

Cut-And-Stitch



Cut step:

*E*stimate posteriors (*E*)



Carnegie Mellon

Cut-And-Stitch

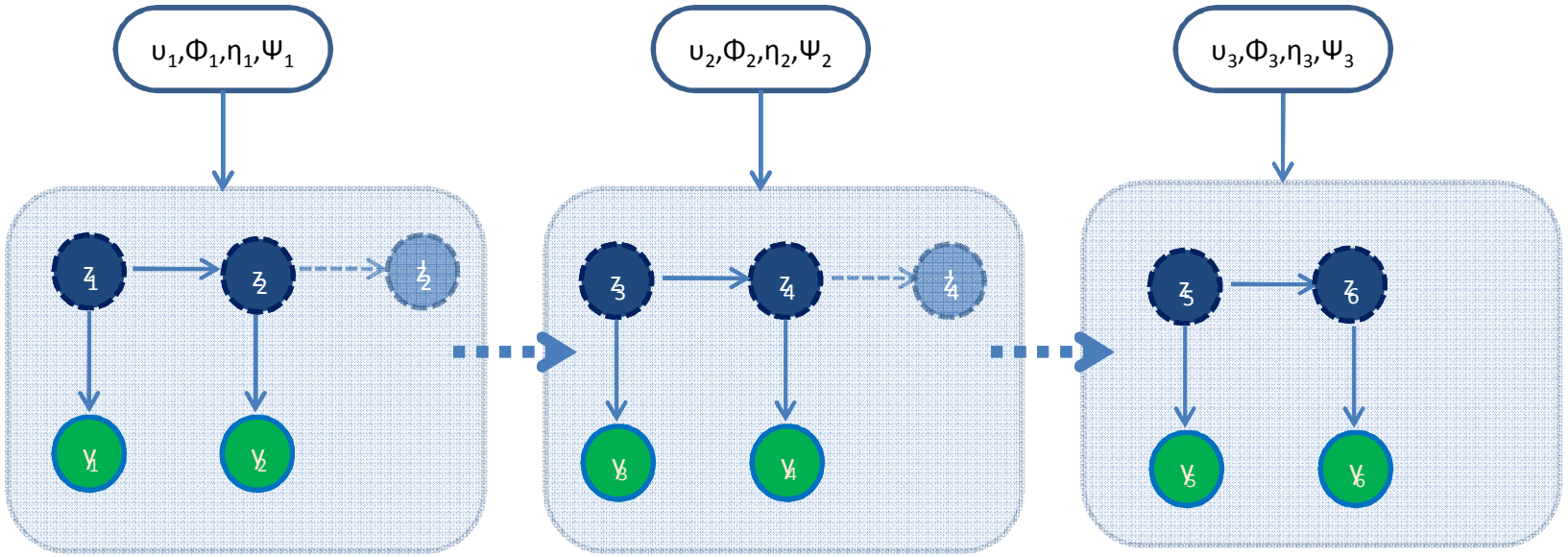
Stitch step:

*C*ollect sufficient statistics for each block (*C*)

$E[z_i]$

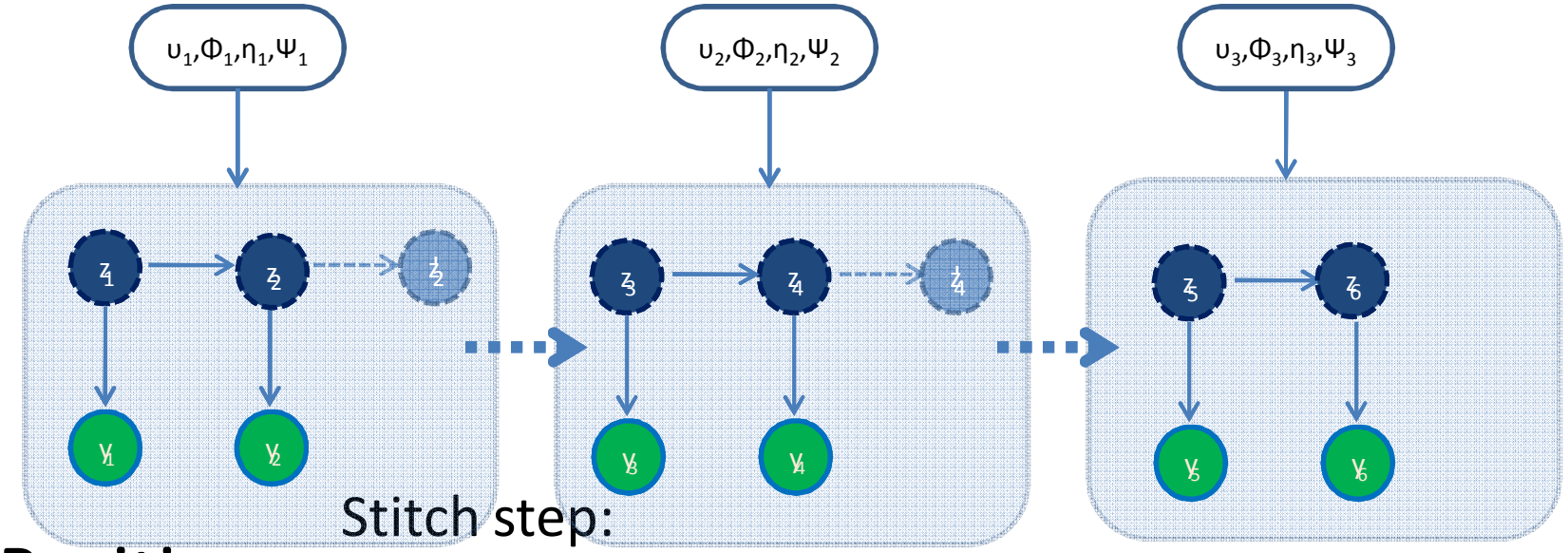
$E[z_i z_i']$

$E[z_{i-1} z_i']$



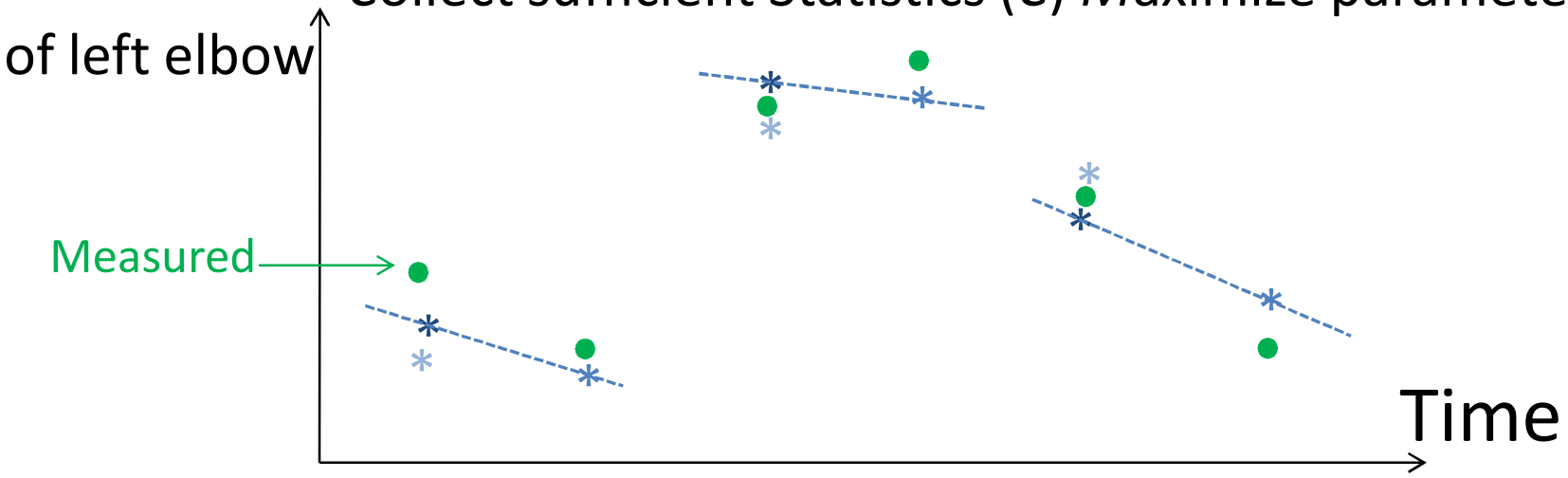
Carnegie Mellon

Cut-And-Stitch



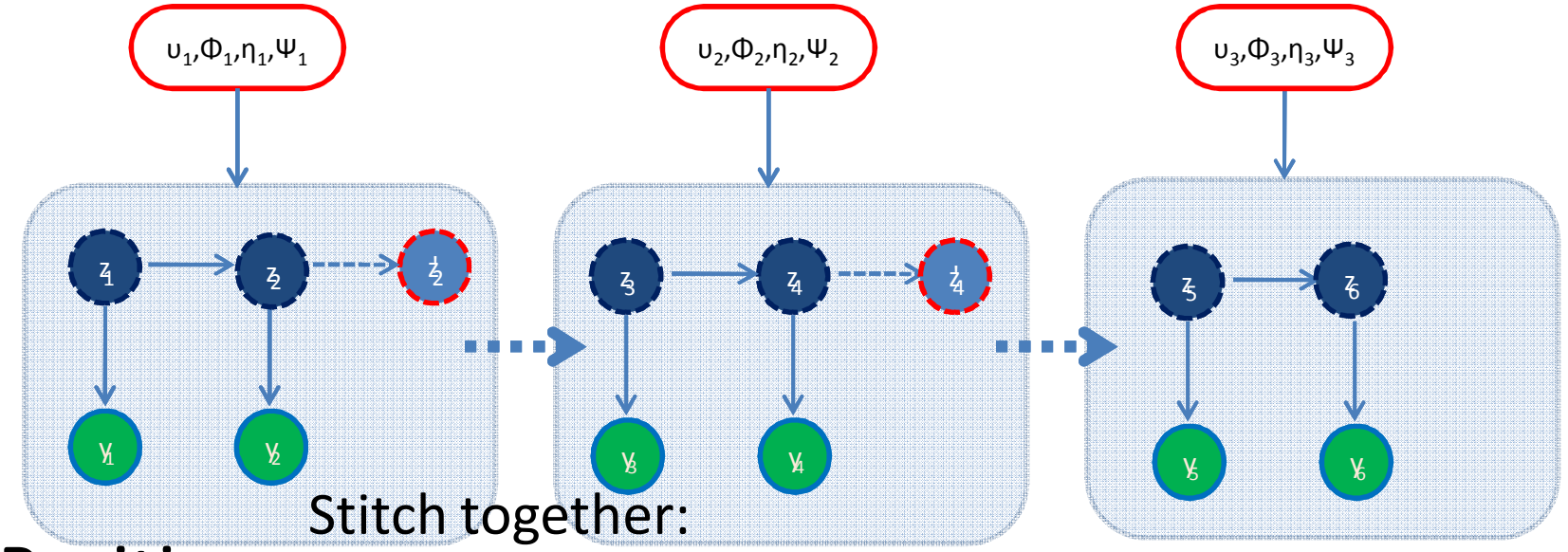
Stitch step:

*C*ollect sufficient Statistics (*C*) *M*aximize parameters (*M*)



Carnegie Mellon

Cut-And-Stitch



Stitch together:

Re-estimate block parameters (*R*)

Position of left elbow

Intuition: exchange messages cross block

Iterate… reconstructed signal

Measured

Time

Carnegie Mellon

Experiments

Q1: How much speed up can we get?

Q2: How good is the reconstuction accuracy?

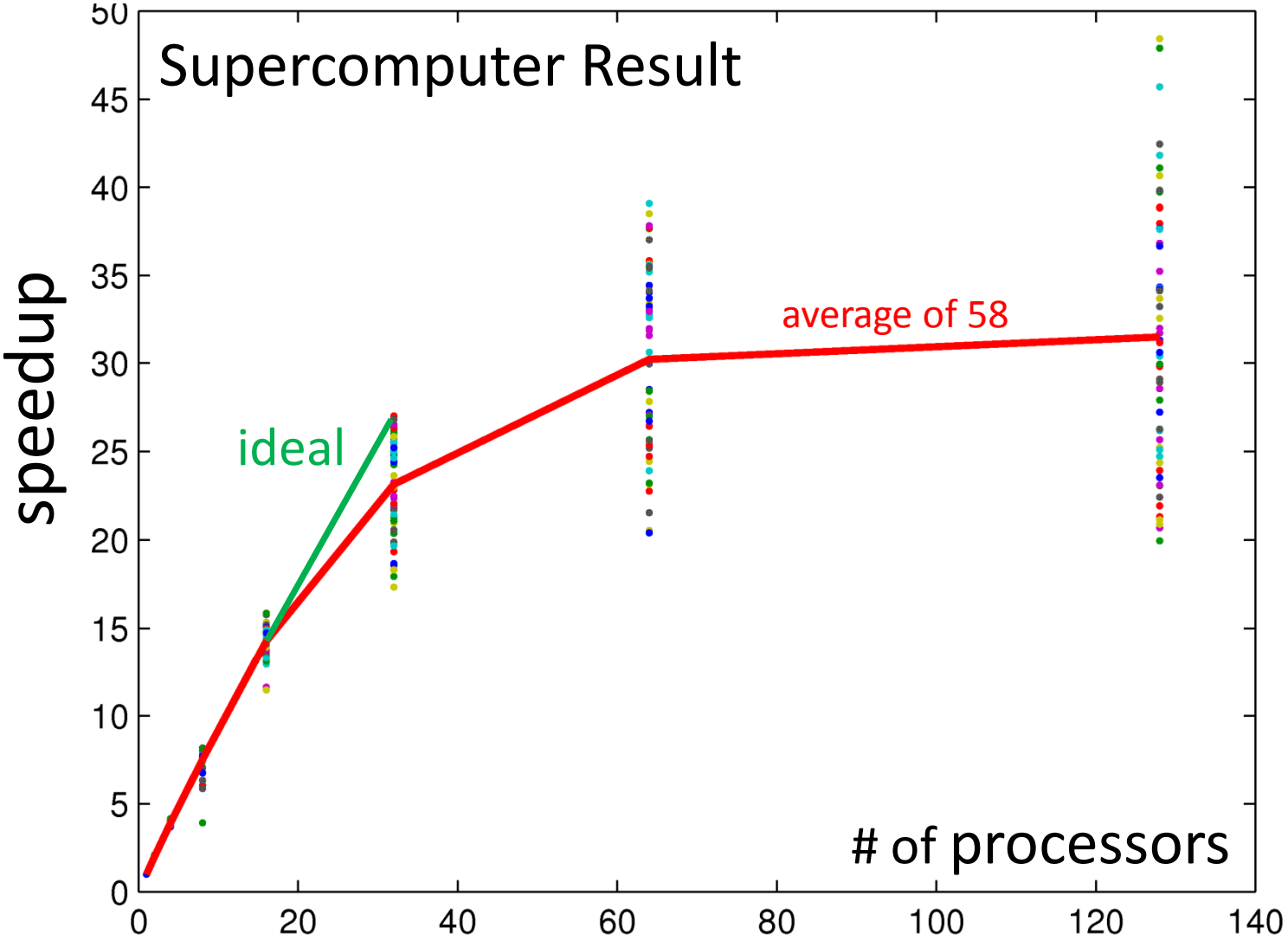
Carnegie Mellon

Experiments

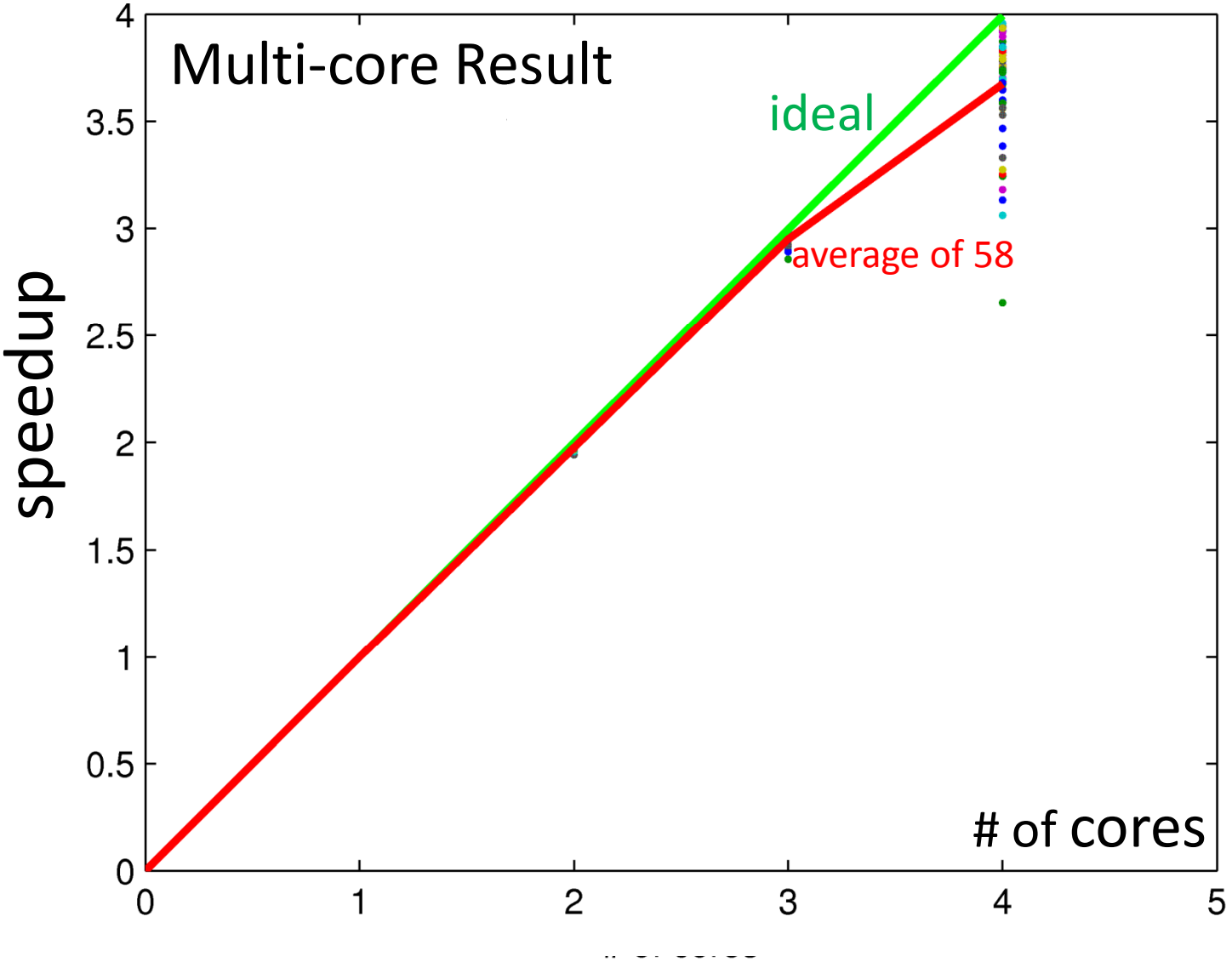
- Dataset:
  - 58 human motion sequences, 200 – 500 frames
  - Each frame with 93 bone positions in body local coordinates
  - http://mocap.cs.cmu.edu
- Setup:
  - Supercomputer: SGI Altix system, distributed shared memory architecture
  - Multi-core desktop: 4 Intel Xeon cores, shared memory
- Task:
  - Learn the dynamics, hidden variables and reconstruct motion

Carnegie Mellon

Q1: How much speed up?

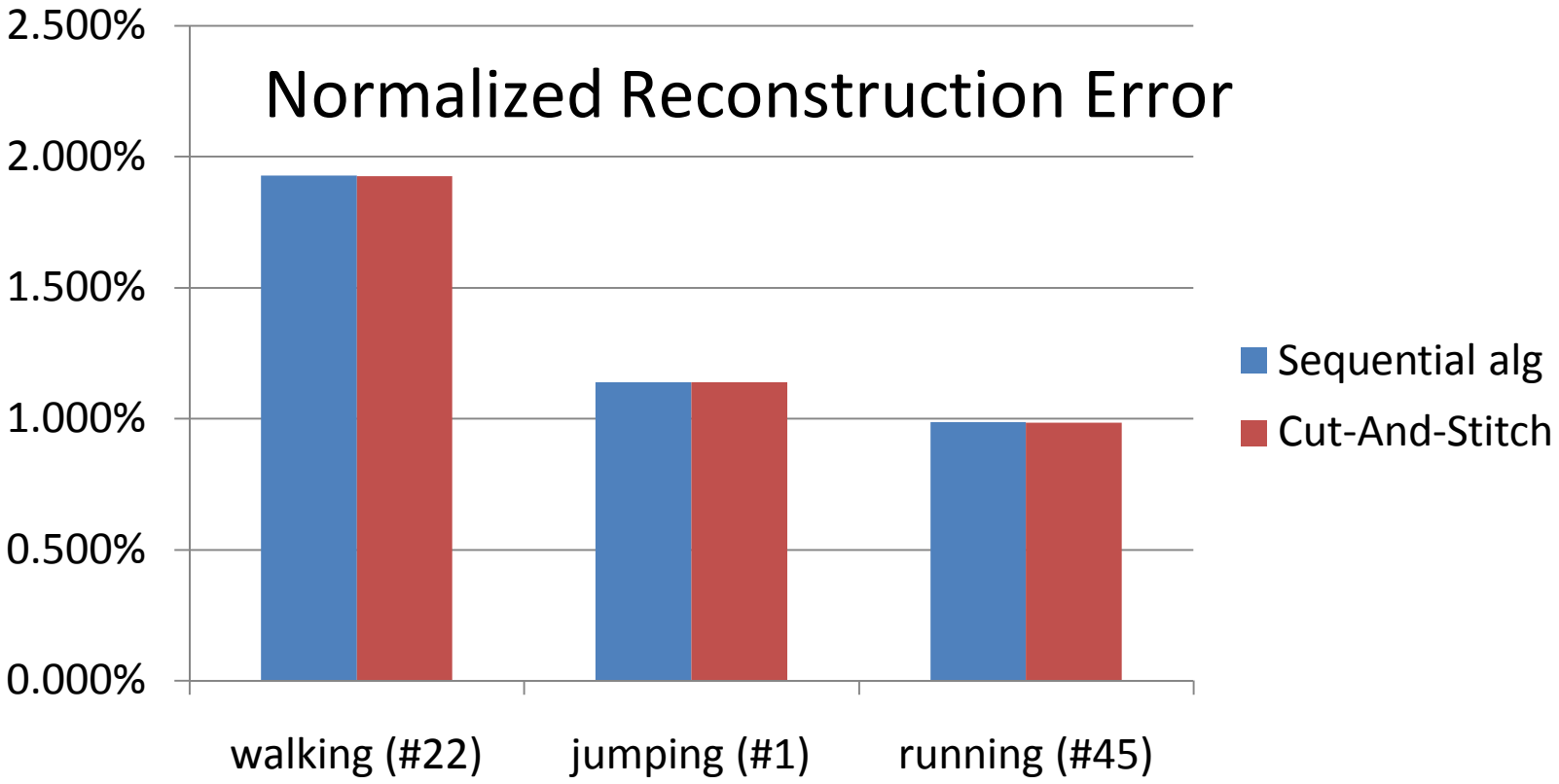


Q1: How much speed up?



Carnegie Mellon

Q2: How good?



Result: ~ IDENTICAL accuracy

Carnegie Mellon

Conclusion & Contributions

General approximate parallel learning algorithm for LDS

- Near linear speed up
- Accuracy (NRE): ~ identical to sequential learning
- Easily extended to HMM and other chain Markovian models

Software (C++ w. openMP) and datasets:
www.cs.cmu.edu/~leili/paralearn

Thank you

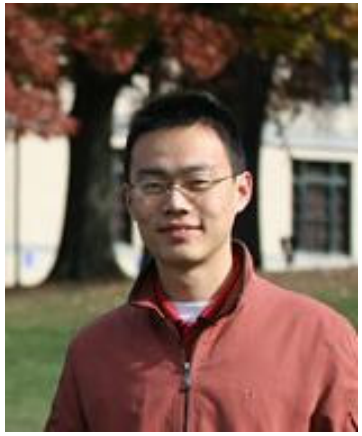
- Questions?

  WELCOME TO POSTER TONIGHT

- Contact info: leili@cs.cmu.edu

Christos Faloutsos

Wenjie Fu



Fan Guo

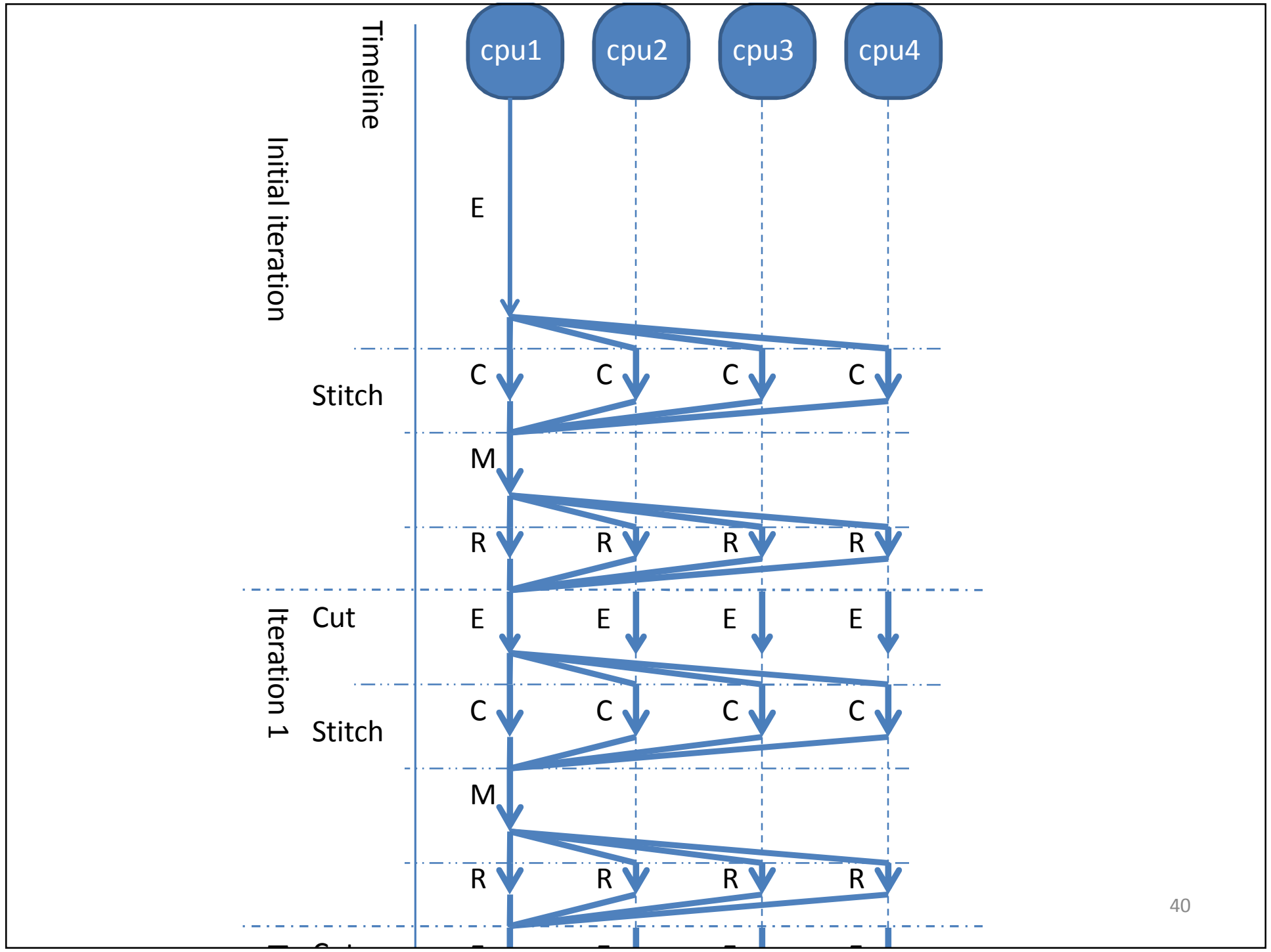
Lei Li

Todd Mowry

Carnegie Mellon

Promising Extensions

- Extension
  - HMM
  - other Markov models (*similar graphical model*)
- Open Problem:
  - Can prove the error bound?


cpu1
cpu2
cpu3
cpu4
Timeline
Initial iteration
E
C
C
C
C
Stitch
M
R
R
R
R
Iteration 1
Cut
E
E
E
E
C
C
C
C
Stitch
M
R
R
R
R

Carnegie Mellon

Approximate Parallel Learning