Transfer Learning by Discovering Latent Task Parameterizations

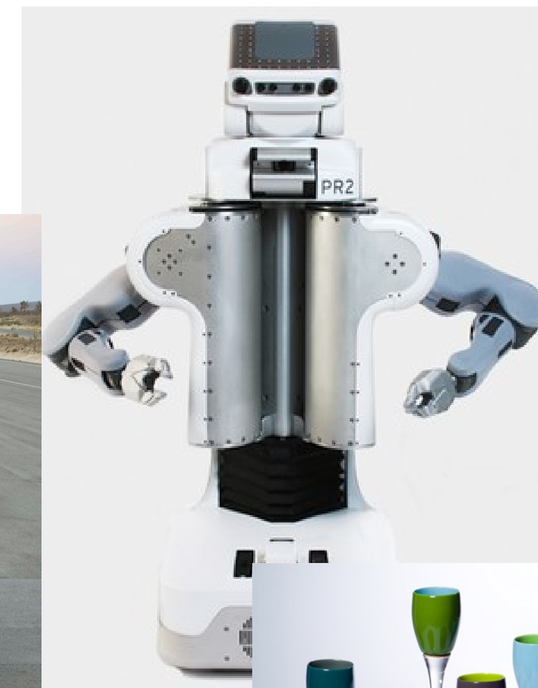
NIPS Workshop on Bayesian Nonparametric Models For Reliable Planning And Decision-Making Under Uncertainty

December 8th 2012

Finale Doshi-Velez
George Konidaris

Motivation: Learning for Control

Real-life problems repeat, but not exactly.

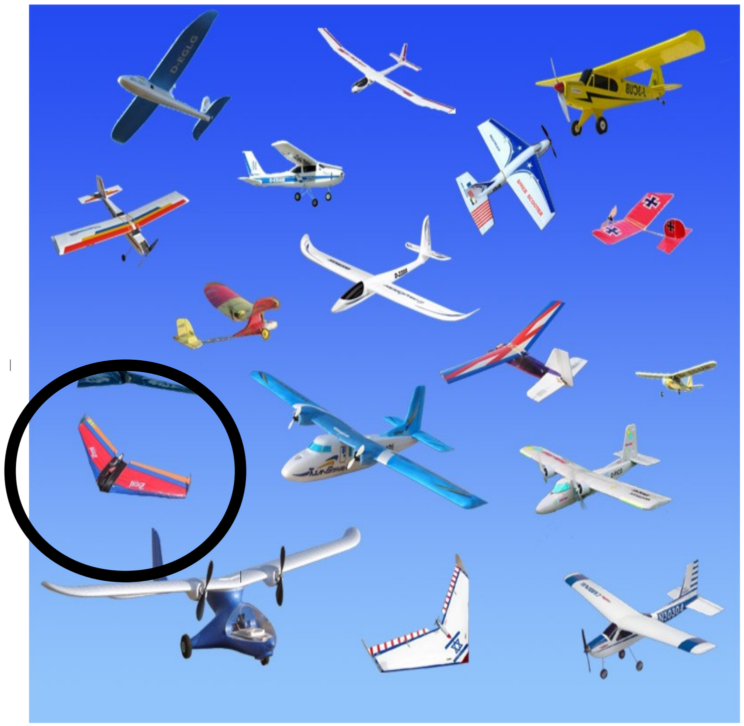


Latent Task Parameterizations

- We model related tasks as a *parametrized family of MDPs.*

Latent Task Parameterizations

- We model related tasks as a *parametrized family of MDPs.*
- Each task is an *instance* obtained by fixing the parameters (we know when this *instance* changes).



Latent Task Parameterizations

- We model related tasks as a *parametrized family of MDPs.*
- Each task is an *instance* obtained by fixing the parameters (we know when this *instance* changes).
- The MDP parameters are never observed: must be inferred from data.

Formalization of the Problem

- Data is a sequence of state transitions:

$$\left(x_n^b, a, \Delta x_n^b\right), b \in \left(1 \ldots B\right)$$

  where b is the “batch” or particular instance

- The dynamics are related through a *shared* set of basis functions and batch-dependent *weights*

$$\Delta x_n^b \sim \sum_k^K w_{kab} f_{ka}\left(x_n^b, \theta\right) + \epsilon_n^b$$

$$\epsilon_n^b \sim N\left(0, \sigma^2\right)$$

Formalization of the Problem

- Data

  whe e

  The weights w_{kb} represent the *minimum amount of information* that we need to learn per instance.

- The *d* set of basis functions and batch-dependent *weights*

$$\Delta x_n^b \sim \sum_k^K w_{kab} f_{ka}\left(x_n^b, \theta\right) + \epsilon_n^b$$

$$\epsilon_n^b \sim N\left(0, \sigma^2\right)$$

Placing Priors

- Data is a sequence of state transitions:

$$\left(x_n^b, a, \Delta x_n^b\right), b \in (1 \ldots B)$$

Batch-specific
Latent parameter
$w_{1ab} = 1$
$w_{kab} \sim N(0, \sigma_w)$

where b is the … particular instance

- The dynamics … through a *shared* set … unctions … batch-dependent *weights*

Finite number
of bases
K ~ Geom(p)

$$\Delta x_n^b \sim \sum_{k}^{K} w_{kab} f_{ka}\left(x_n^b, \theta\right)$$

Shared
basis functions
f_{ka} ~ GP(θ)

$$\epsilon_n^b \sim N\left(0, \sigma^2\right)$$

... this is essentially the Semiparametric Latent Factor Model (Teh, Seegar, Jordan 2005)

Approach

- Given batch data from many instances, we learn the number of bases (K) and define a posterior over the functions $f_{ka}()$.
- Then, given data from a new instance b', we only need to filter over the batch-specific latent parameters $w_{kab'}$.

A Few Details: Marginal Likelihood

Batch training: Integrating out the bases $f_{ka}()$ for each action *a*, we get the marginal likelihood

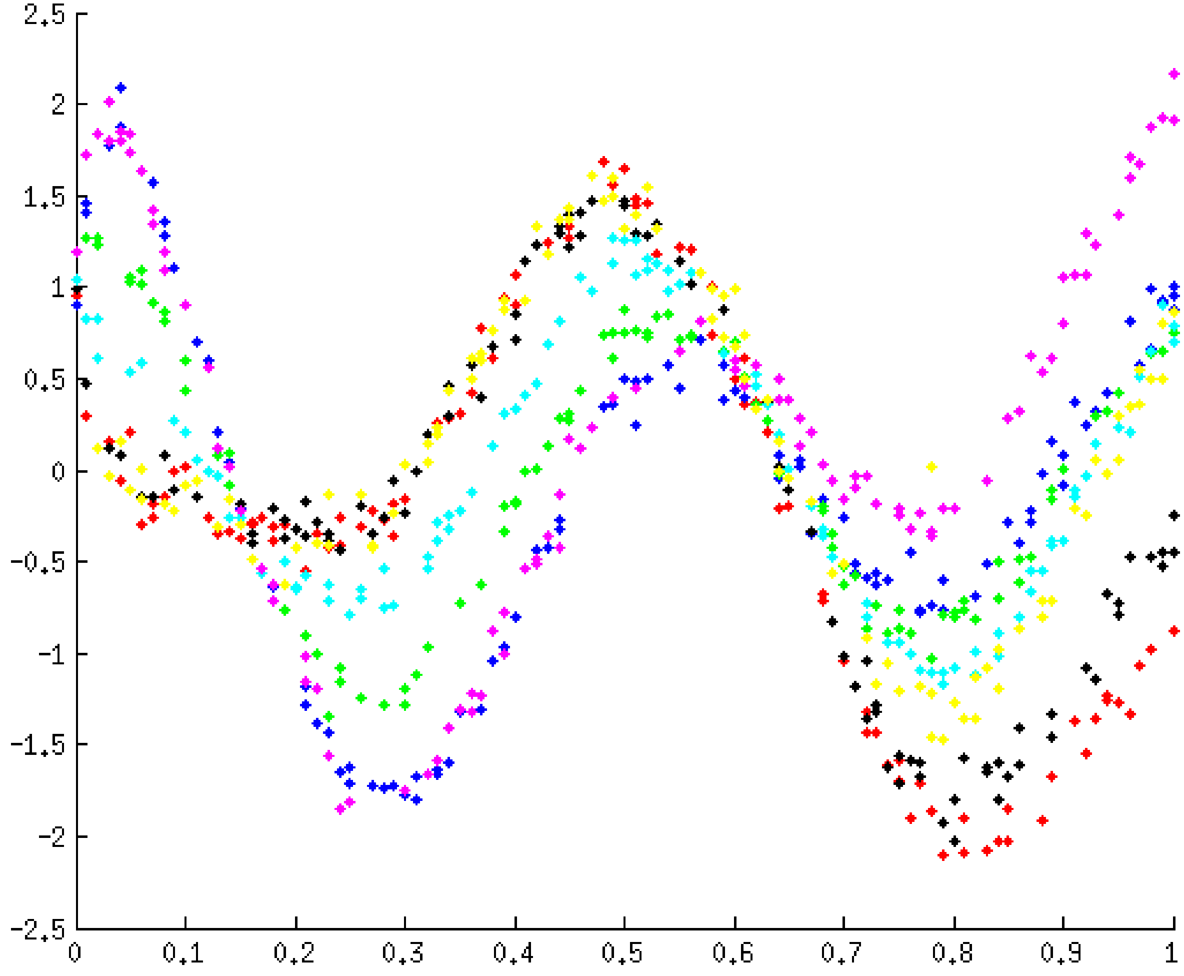
$$P(\Delta X | W, X) = N(0, K(X, X) \odot (A^T W^T W A))$$

where

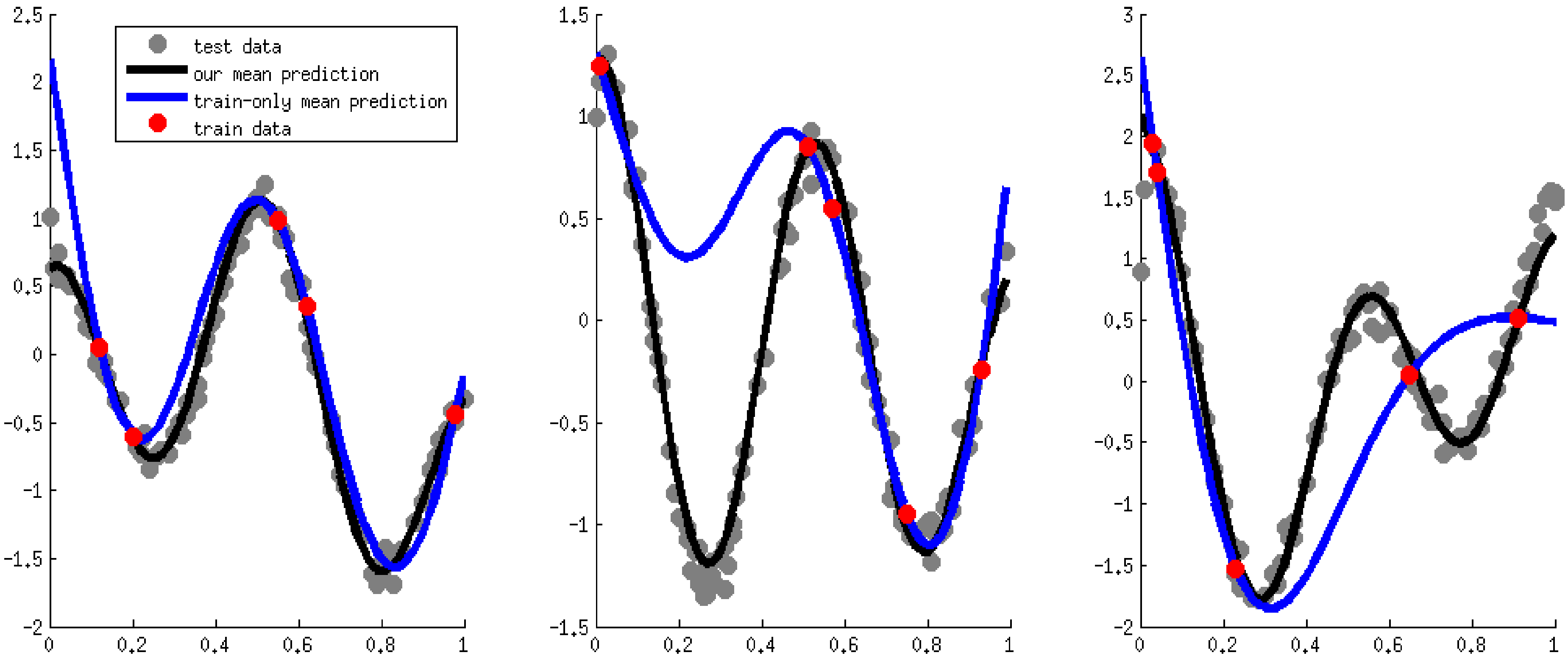
- ΔX is an *N*x*1* vector of the differences
- *W* is the *K*x*B* matrix of the weights
- *A(b, n) = 1* if the *n*th data point came from batch *b*; 0 otherwise.

Inference: MH on W, RJ-MCMC on K.

Toy Example: 1D functions Training Data



Toy Example: Test Time



Quick Interlude: Why GPs?

(yes, we did our homework)

We did an initial exploration of what form of basis functions could be used to approximate a *single* batch Cartpole/Acrobot well.

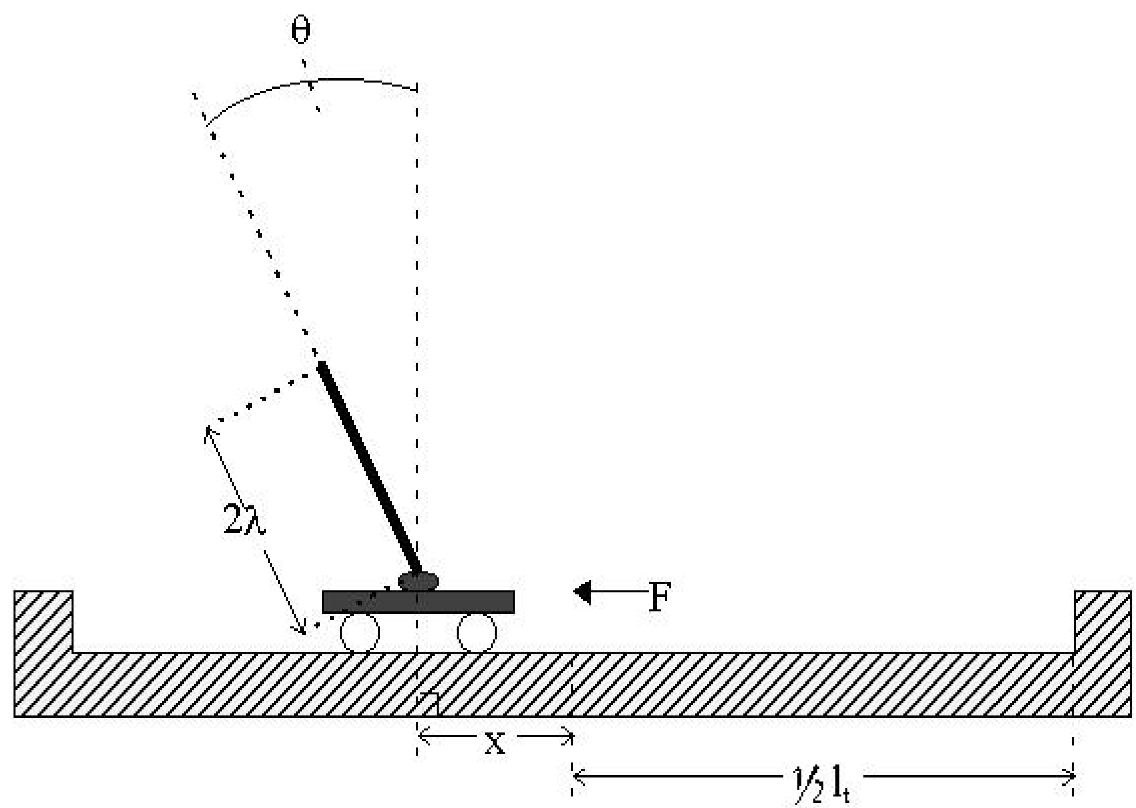
- Obviously, if we include terms from the physics, we could make very good predictions.
- We found that we needed 8-10 Fourier bases (and even more polynomial bases) to get decent predictions... perhaps too general.
- GPs let us create application-specific bases.

Cart-Pole Example

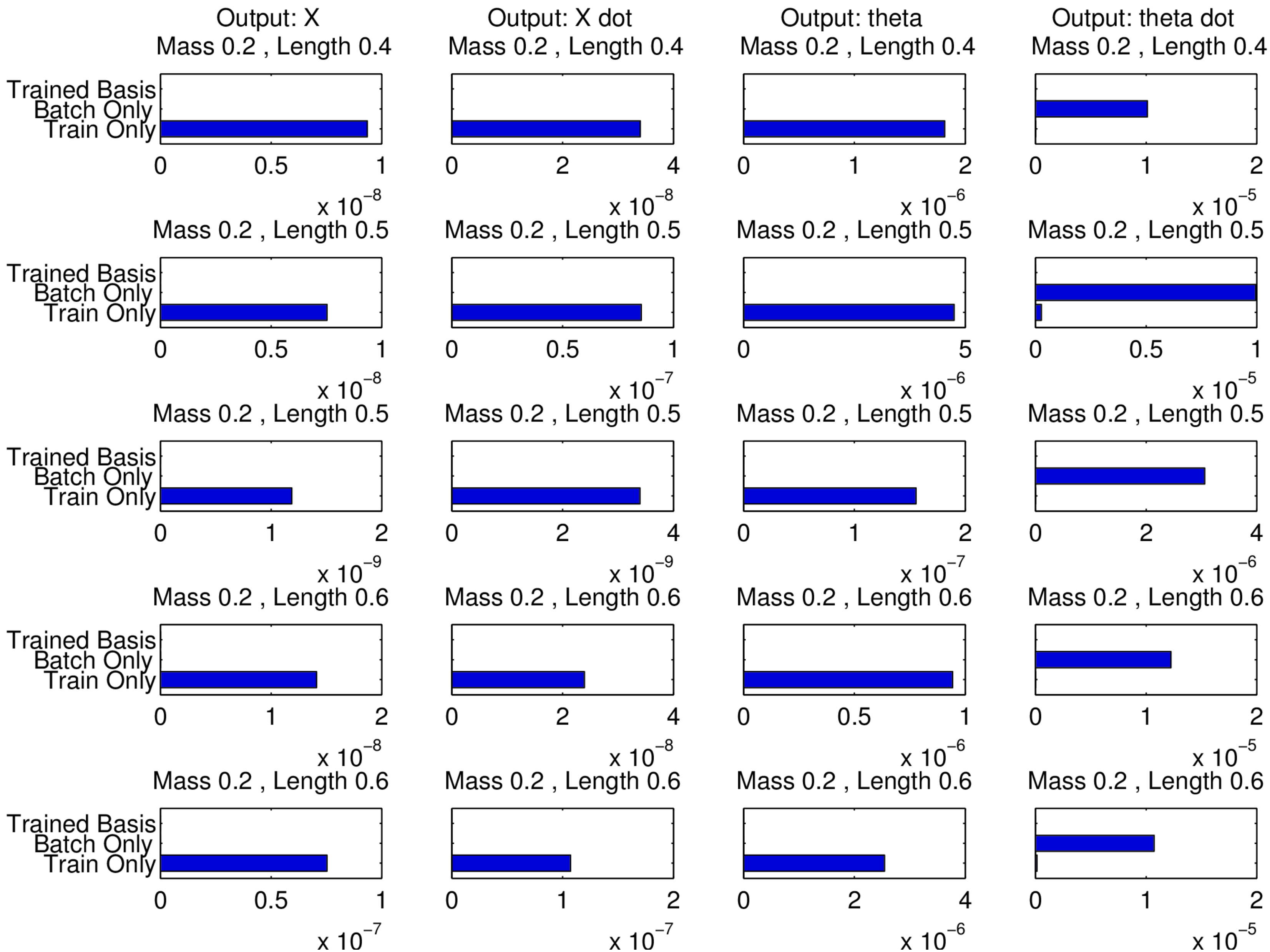
4-D domain:

- Inputs: $x, \dot{x}, \theta, \dot{\theta}$
- Outputs: $\dot{x}, \ddot{x}, \dot{\theta}, \ddot{\theta}$

Instances vary the weight and the length of the pole.

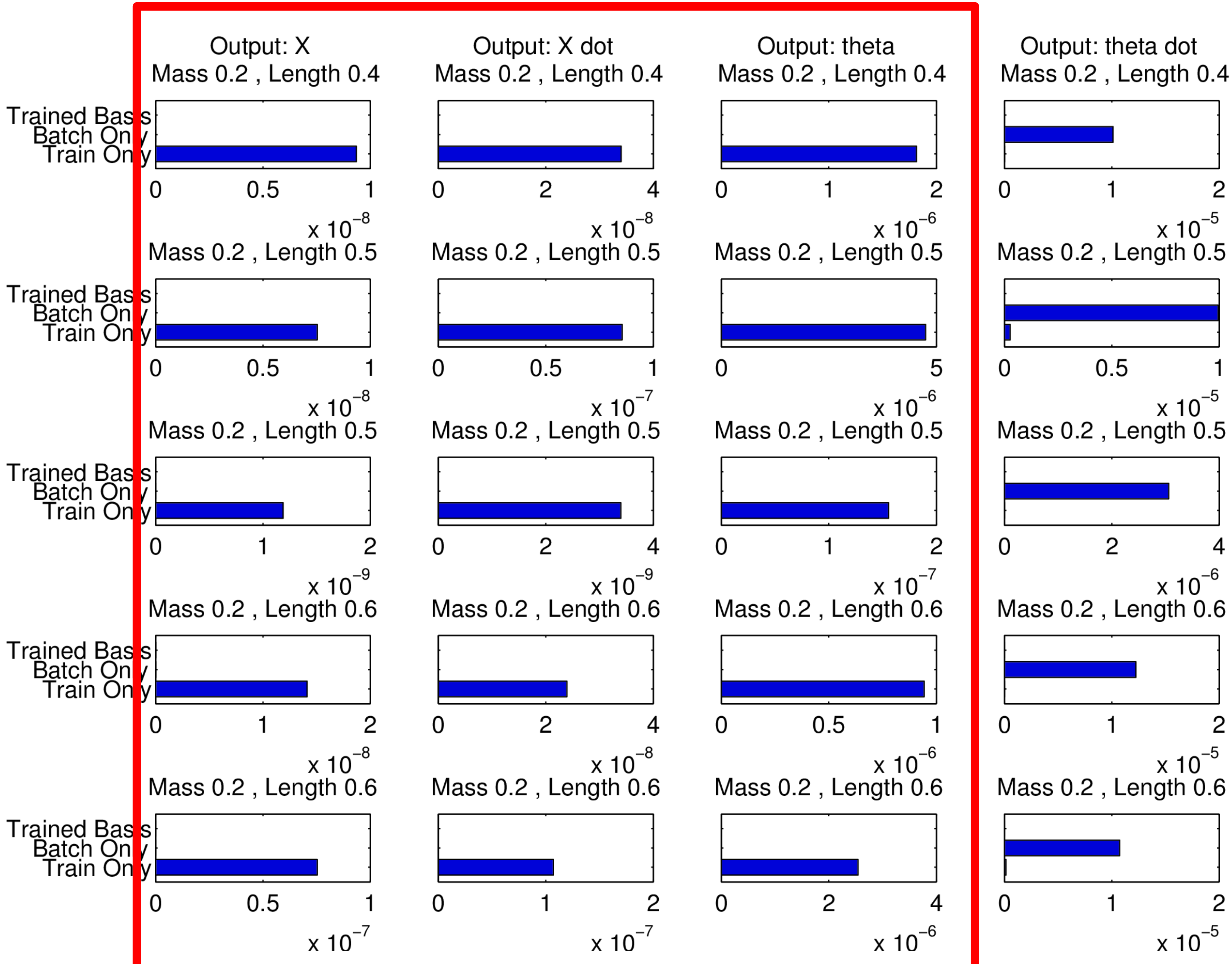


Varying Length



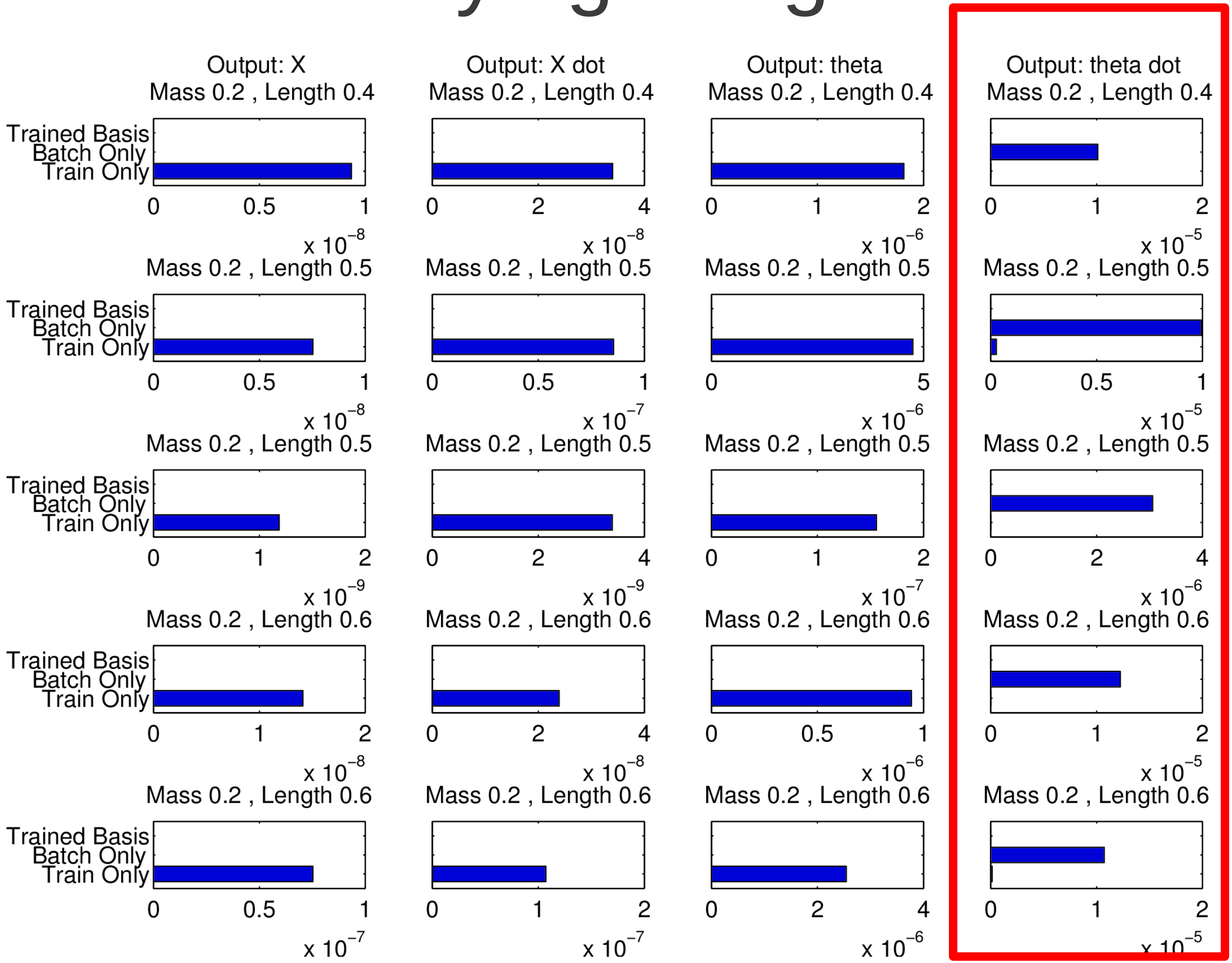
Varying Length

No extra bases added

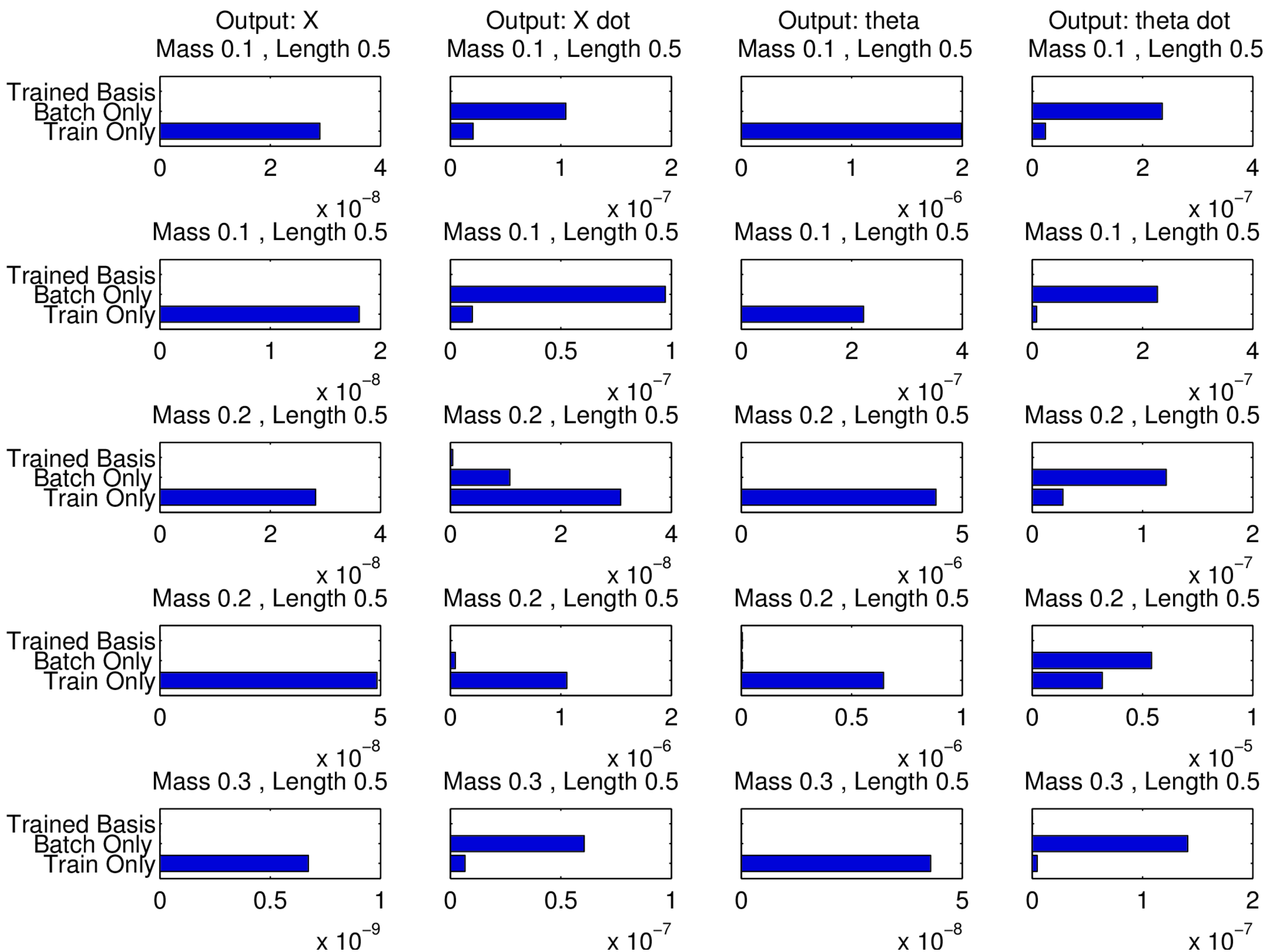


Varying Length

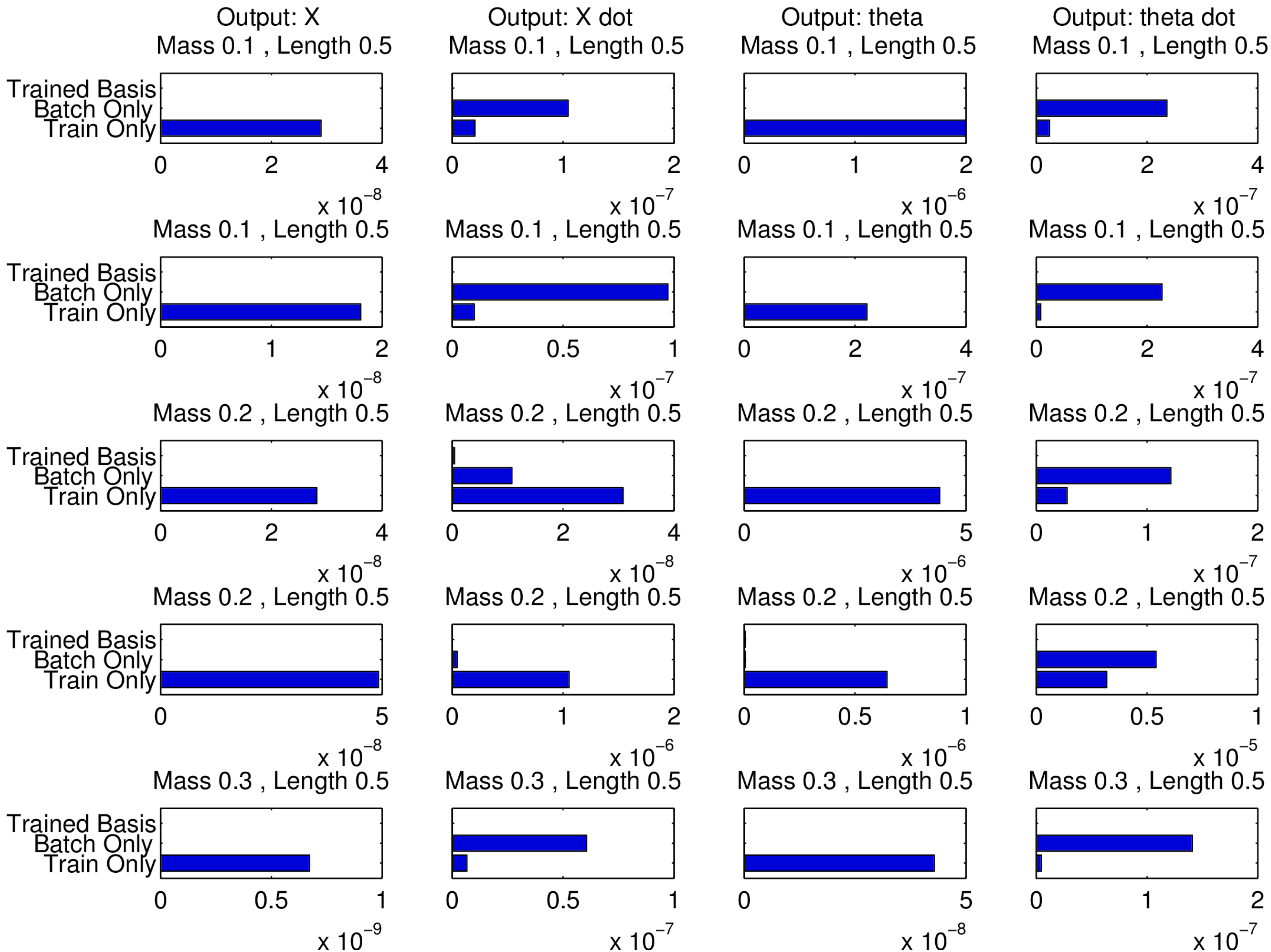
One basis added



Varying Mass

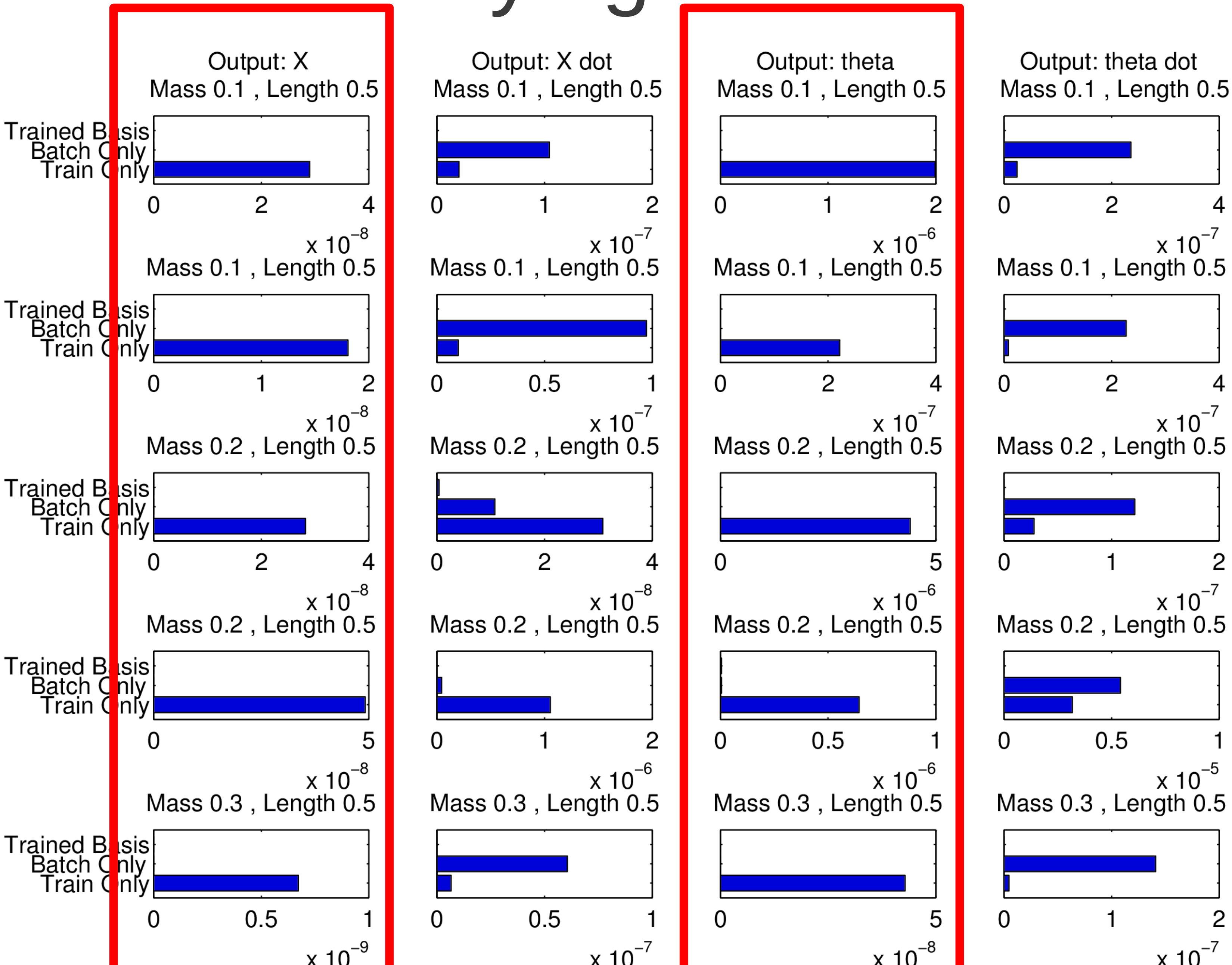


Varying Mass

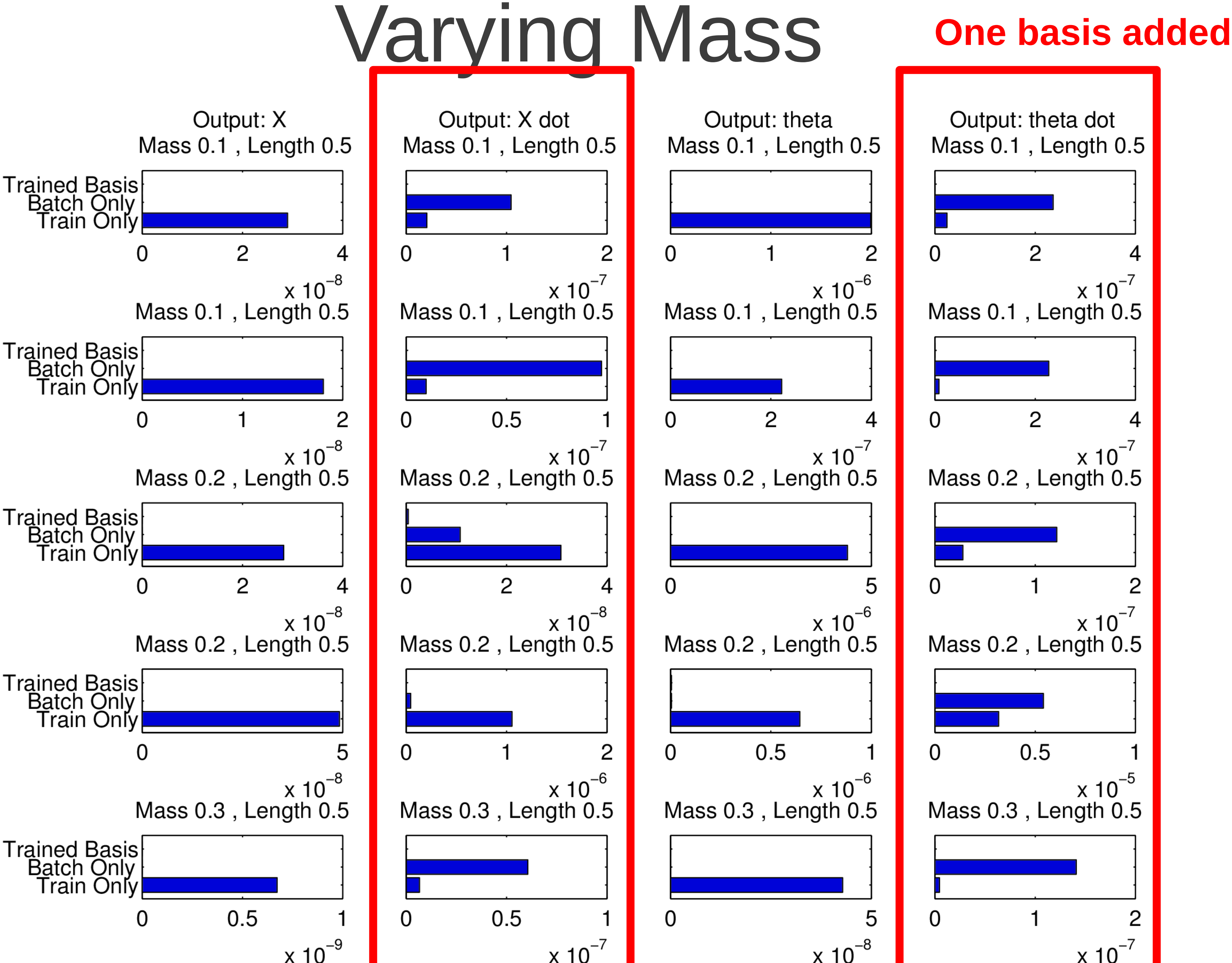


Varying Mass

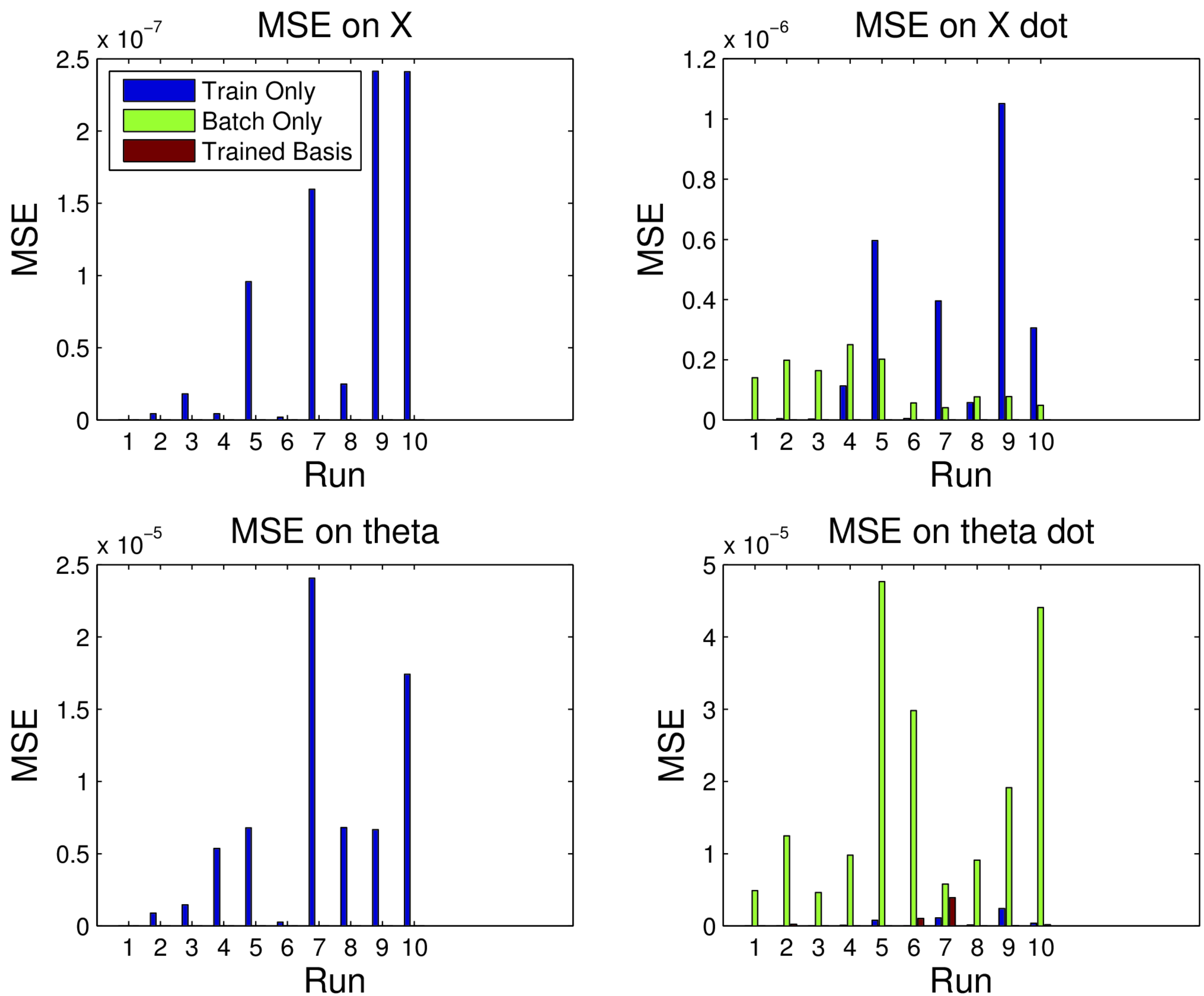
No bases added



Varying Mass



Varying Mass and Length: 3 Bases Added



Conclusions and Future Work

- We demonstrated our latent parametrization approach on a sample problem, cartpole.
- Currently modifying approach so that the latent parameters are shared across all actions and outputs, making inference more efficient.
- Next steps: Close the control loop by (1) pre-computing belief-space policies and (2) filtering over weights for a new instance.

HiP-MDP

Laying out the model...

- We can model this as a HiP-MDP:

$$< S, A, \Theta, T, R, \gamma, P_\Theta >$$

where

S: state space

A: action set

$T(s'|s,a,\theta)$: dynamics

$R(s'|s,a,\theta)$: reward function

Θ: parameter space

P_Θ: parameter pdf

γ : discount factor

The Details: Inference

- For the weight values w_{kb} we use MH, with acceptance threshold:

$$a = \min(1, \frac{P(Y|W')P(w'_{kb}|W_{-kb})}{P(Y|W)P(w_{kb}|W_{-kb})})$$

- where

$$P(w_{kb}|W_{-kb}) = N(w'_{kb}; 0, \sigma_w^2)$$