Incremental Learning of Robot Dynamics with Constant Complexity

Machine Learning Day @ IIT

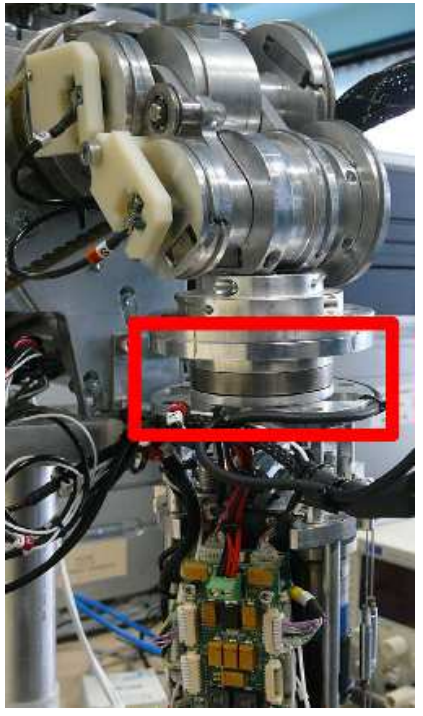
Arjan Gijsberts

June 8th, 2010

Robotics, Brain and Cognitive Sciences

Problem Setting: Inverse Dynamics

Given arm configuration $[\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}]$, estimate internal forces and torques $\left[F_{x,y,z}, \tau_{x,y,z}\right]$.



Purpose:

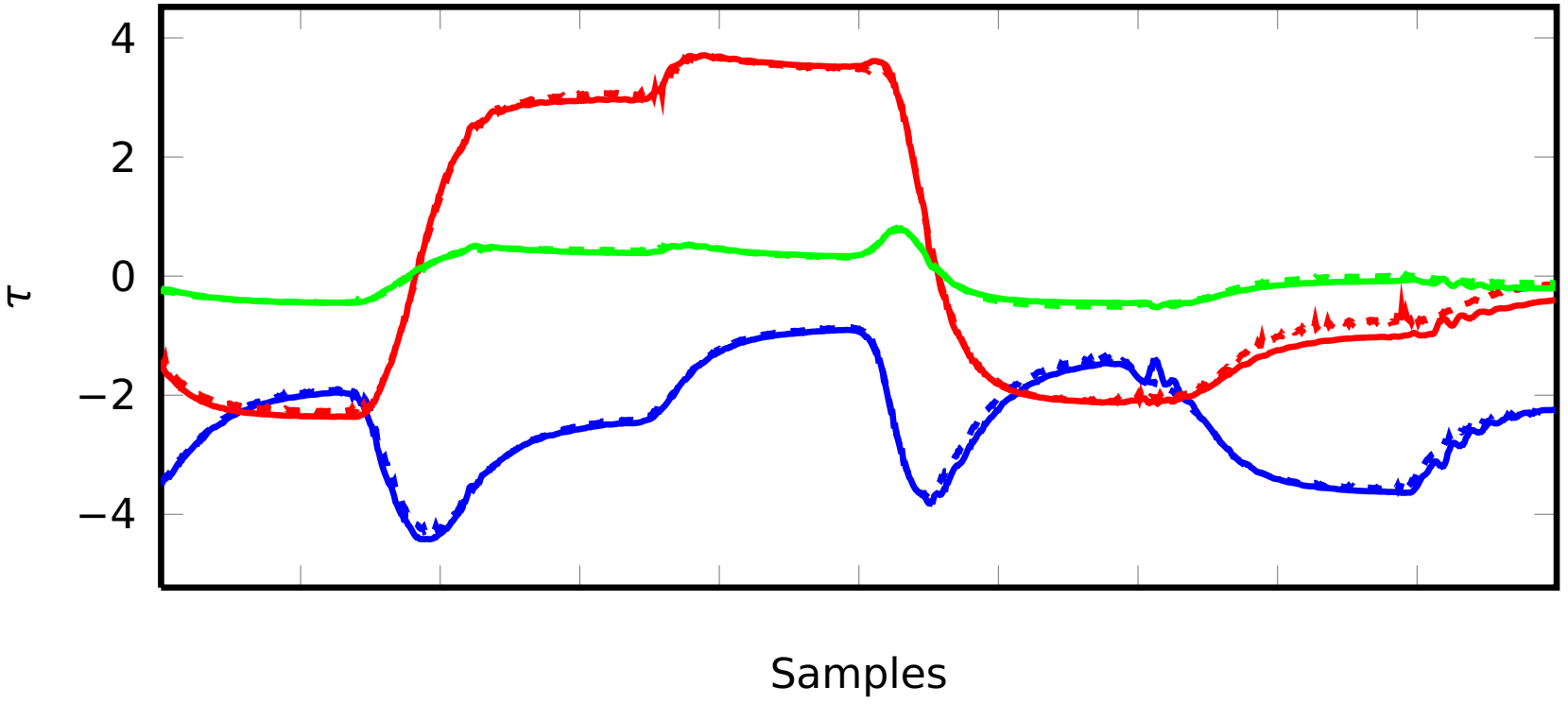
- Compliant control
- Contact detection (cf. $F_E = F - F_I$)

Context:

- Non-linearities (e.g., elasticity)
- Continuous sampling @ 50Hz

Batch Results

Inverse dynamics can be learned: on our dataset, ML algorithms get NMSE < 1% (Fumagalli *et al.*, 2010).



Incremental Learning

We want to adapt the model *incrementally* and *continuously* during the robot's operation, so that we can cope with concept drift (e.g., sensor drift, changed model during tool use).

We thus want a method that

- generalizes well
- is non-linear
- is incremental
- is practical
- can run forever ($\mathcal{O}(1)$ per update)

Incremental Learning

We want to adapt the model *incrementally* and *continuously* during the robot's operation, so that we can cope with concept drift (e.g., sensor drift, changed model during tool use).

We thus want a method that

- generalizes well
- is non-linear
- is incremental
- is practical
- can run forever ($\mathcal{O}(1)$ per update)

Kernel Methods

O(1) Kernel Methods

- Fixed budget B for support set
  - replace random, smallest coefficient, oldest (FIFO), more advanced...
- Project linear dependent vectors (with tolerance ϵ) onto each other
- Approximate kernel with a finite D-dimensional feature mapping
  - no more need for kernel expansion
  - exact mapping for polynomial kernel

Random Features (1)

Recall:

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle \qquad \text{where } \phi(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathcal{H}$$

RKHS $\mathcal{H}$ is infinite dimensional for some kernels (e.g., RBF); we cannot compute $\phi(\mathbf{x})$.

What if we had a *random* mapping $z(\mathbf{x})$, such that

$$k(\mathbf{x}, \mathbf{y}) = \mathbb{E}\left[\langle z(\mathbf{x}), z(\mathbf{y}) \rangle \right]$$

(Rahimi & Recht, 2008)

Random Features (2)

Bochner's Theorem: *A continuous kernel* $k(\mathbf{x}-\mathbf{y})$ *on* $\mathbb{R}^d$ *is positive definite iff* $k(\delta)$ *is the Fourier transform of a non-negative measure* $p(\omega)$.

$$k(\mathbf{x},\mathbf{y}) = \int_{\mathbb{R}^d} e^{-i\omega^{\mathsf{T}}(\mathbf{x}-\mathbf{y})} p(\omega)\,\mathrm{d}\omega$$

Random Features (2)

Bochner's Theorem: *A continuous kernel* $k(\mathbf{x}-\mathbf{y})$ *on* $\mathbb{R}^d$ *is positive definite iff* $k(\delta)$ *is the Fourier transform of a non-negative measure* $p(\omega)$.

$$k(\mathbf{x},\mathbf{y}) = \int_{\mathbb{R}^d} e^{-i\omega^{\mathsf{T}}(\mathbf{x}-\mathbf{y})} p(\omega)\,\mathrm{d}\omega$$

$$= \mathbb{E}\left[e^{i\omega^{\mathsf{T}}\mathbf{x}} e^{i\omega^{\mathsf{T}}\mathbf{y}^H}\right] \qquad \text{iff} \int p(\omega)\,\mathrm{d}\omega = 1$$

Random Features (2)

Bochner's Theorem: *A continuous kernel* $k(\mathbf{x}-\mathbf{y})$ *on* $\mathbb{R}^d$ *is positive definite iff* $k(\delta)$ *is the Fourier transform of a non-negative measure* $p(\omega)$.

$$
\begin{aligned}
k(\mathbf{x},\mathbf{y}) &= \int_{\mathbb{R}^d} e^{-i\omega^{\mathsf{T}}(\mathbf{x}-\mathbf{y})} p(\omega)\,\mathrm{d}\omega \\
&= \mathbb{E}\left[e^{i\omega^{\mathsf{T}}\mathbf{x}} e^{i\omega^{\mathsf{T}}\mathbf{y}^H}\right] && \text{iff } \int p(\omega)\,\mathrm{d}\omega = 1 \\
&= \mathbb{E}\left[\langle z_\omega(\mathbf{x}), z_\omega(\mathbf{y})\rangle\right] && \text{as } k(\delta), p(\omega) \in \mathbb{R}^d \\
z_\omega(\mathbf{x}) &= \sqrt{2}\cos(\langle \omega, \mathbf{x}\rangle + b) && p(b) \sim U(0, 2\pi)
\end{aligned}
$$

Random Features (2)

Bochner's Theorem: *A continuous kernel* $k(\mathbf{x}-\mathbf{y})$ *on* $\mathbb{R}^d$ *is positive definite iff* $k(\delta)$ *is the Fourier transform of a non-negative measure* $p(\omega)$.

$$k(\mathbf{x},\mathbf{y}) = \int_{\mathbb{R}^d} e^{-i\omega^\mathsf{T}(\mathbf{x}-\mathbf{y})} p(\omega)\,\mathrm{d}\omega$$

$$= \mathbb{E}\left[e^{i\omega^\mathsf{T}\mathbf{x}} e^{i\omega^\mathsf{T}\mathbf{y}^H}\right] \qquad \text{iff} \int p(\omega)\,\mathrm{d}\omega = 1$$

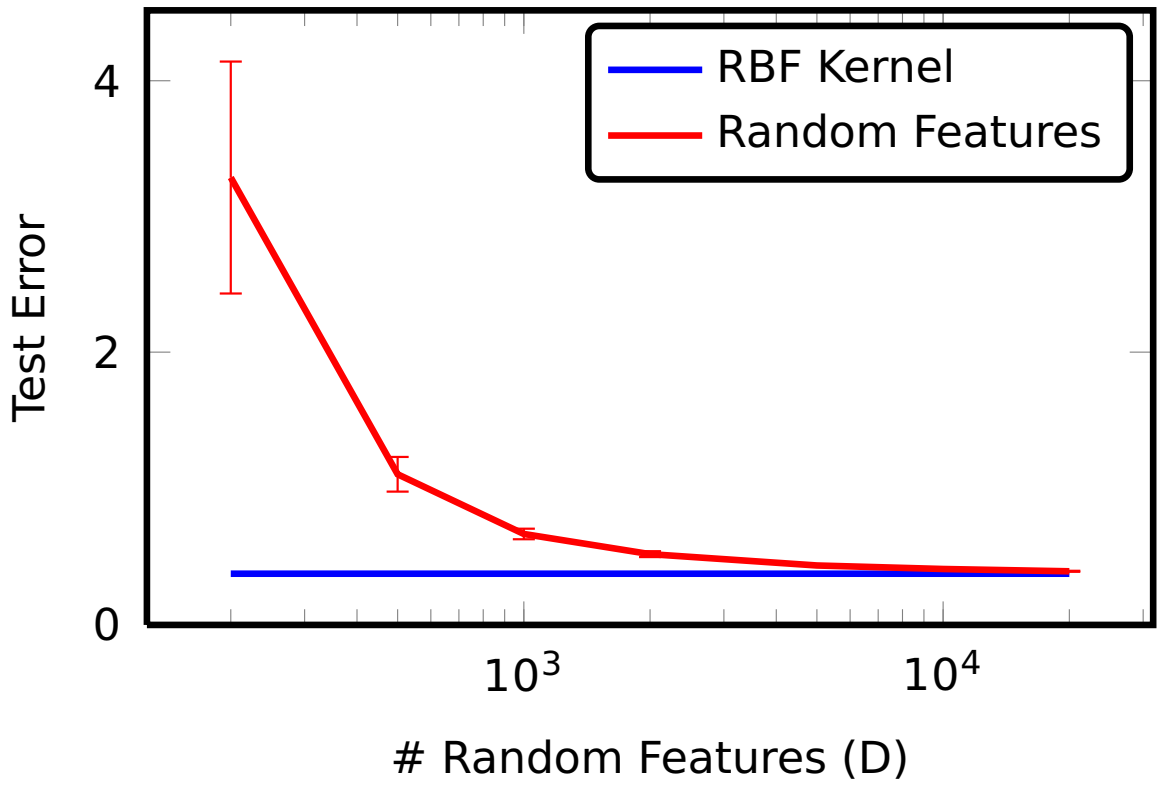
$$= \mathbb{E}\left[\langle z_\omega(\mathbf{x}), z_\omega(\mathbf{y})\rangle\right] \qquad \text{as } k(\delta), p(\omega) \in \mathbb{R}^d$$

$$z_\omega(\mathbf{x}) = \sqrt{2}\cos(\langle\omega,\mathbf{x}\rangle + b) \qquad p(b) \sim U(0, 2\pi)$$

$$p(\omega) \sim \mathcal{N}(0, 2\gamma\mathbf{I}) \qquad \text{for } k(\mathbf{x},\mathbf{y}) = e^{-\gamma\|\mathbf{x}-\mathbf{y}\|^2}$$

Performance vs # Random Features

$$\left\| \hat{f} - f \right\|_\infty \in \mathcal{O}\left(\frac{1}{\sqrt{D}} \right)$$



Regularized Least Squares

Given inputs $\mathbf{X} = \left[\mathbf{x}_1^\mathsf{T}, \ldots, \mathbf{x}_m^\mathsf{T}\right]^\mathsf{T} \in \mathbb{R}^{m \times d}$ and outputs $\mathbf{y} = [y_1, \ldots, y_m]^\mathsf{T} \in \mathbb{R}^m$, find function $f(\mathbf{x})$ such that the empirical error and the capacity of the function are minimized.

Regularized Least Squares

Given inputs $\mathbf{X} = \left[\mathbf{x}_1^\mathsf{T}, \ldots, \mathbf{x}_m^\mathsf{T}\right]^\mathsf{T} \in \mathbb{R}^{m \times d}$ and outputs $\mathbf{y} = [y_1, \ldots, y_m]^\mathsf{T} \in \mathbb{R}^m$, find function $f(\mathbf{x})$ such that the empirical error and the capacity of the function are minimized.

$$\min_{\mathbf{w}} J(\mathbf{w}, \lambda) = \frac{1}{2} \sum_{i=1}^{m} (y_i - f(\mathbf{x}_i))^2 + \lambda \frac{1}{2} \|f\|^2$$

Regularized Least Squares

Given inputs $\mathbf{X} = \left[\mathbf{x}_1^\mathsf{T}, \ldots, \mathbf{x}_m^\mathsf{T}\right]^\mathsf{T} \in \mathbb{R}^{m \times d}$ and outputs $\mathbf{y} = [y_1, \ldots, y_m]^\mathsf{T} \in \mathbb{R}^m$, find function $f(\mathbf{x})$ such that the empirical error and the capacity of the function are minimized.

$$\min_{\mathbf{w}} J(\mathbf{w}, \lambda) = \frac{1}{2}\sum_{i=1}^{m}(y_i - f(\mathbf{x}_i))^2 + \lambda \frac{1}{2}\|f\|^2$$

Given a feature map $\phi : \mathbb{R}^d \to \mathbb{R}^D$, we can substitute $\mathbf{X}$ with $\mathbf{\Phi} = \left[\phi(\mathbf{x}_1)^\mathsf{T}, \ldots, \phi(\mathbf{x}_m)^\mathsf{T}\right]^\mathsf{T}$.

Incremental RLS (1)

We want to update $\mathbf{w}_t$ efficiently at arrival of a new sample $\mathbf{x}_t$. This is easy using the Sherman-Morrison formula:

$$\begin{aligned}\mathbf{w}_t &= \left(\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^\mathsf{T}\mathbf{y} = \mathbf{A}_t^{-1}\mathbf{b}_t \\ &= \left(\mathbf{A}_{t-1} + \mathbf{x}\mathbf{x}^\mathsf{T}\right)^{-1}(\mathbf{b}_{t-1} + \mathbf{x}_t y_t) \\ &= \left(\mathbf{A}_{t-1}^{-1} - \frac{\mathbf{A}_{t-1}^{-1}\mathbf{x}\mathbf{x}^\mathsf{T}\mathbf{A}_{t-1}^{-1}}{1 + \mathbf{x}^\mathsf{T}\mathbf{A}_{t-1}^{-1}\mathbf{x}}\right)(\mathbf{b}_{t-1} + \mathbf{x}_t y_t)\end{aligned}$$

Similar to Recursive Least Squares in Adaptive Filtering.

Incremental RLS (2)

Two possible issues:

(1) Sherman-Morrison is numerically unstable.

Solution: **A** is symmetric positive definite, we can use rank-1 Cholesky updates instead.

(2) The relative amount of regularization decreases.

Solution: Not really a problem: the more samples, the less likely it is that **A** is ill-conditioned as $D \ll m$.

Vector Valued Outputs

In our case, we want to predict an output vector $\mathbf{y} \in \mathbb{R}^p$. Given identical hyperparameters λ and γ, we can solve for all outputs concurrently:

$$f(\mathbf{x}) = \langle \mathbf{W}, \mathbf{x} \rangle \qquad \text{where } \mathbf{W} \in \mathbb{R}^{d \times p}$$

$$\mathbf{W} = \underbrace{\left(\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda \mathbf{I}\right)^{-1}}_{d \times d} \underbrace{\mathbf{X}^\mathsf{T}\mathbf{Y}}_{d \times p} \qquad \text{where } \mathbf{Y} \in \mathbb{R}^{m \times p}$$

More interesting work on multi-task learning going on recently!

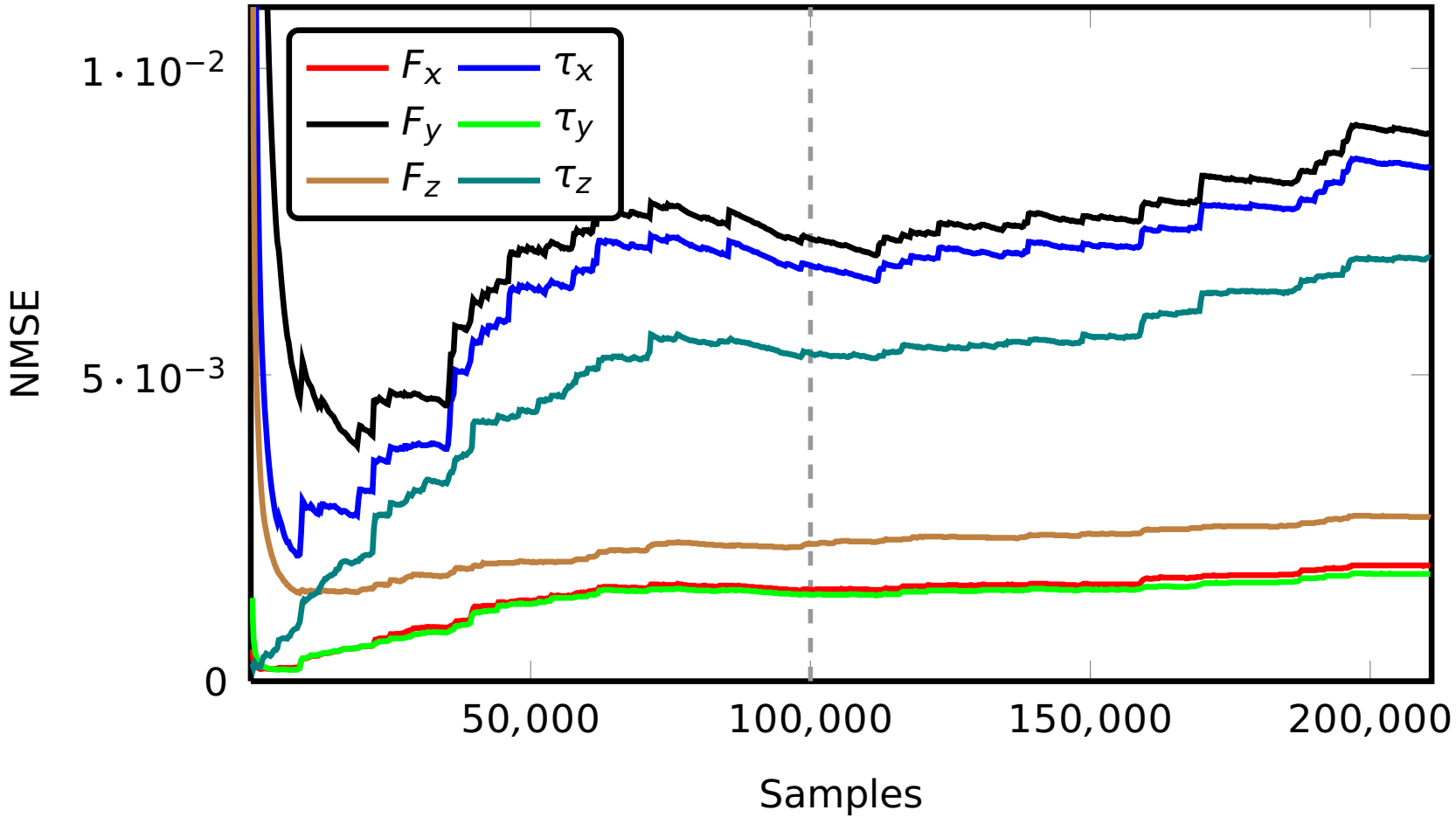
Experimental Setup

- Arm dynamics of James' (4 joints)
- 210997 ordered samples, split as 50000 (training), 50000 (validation) and 110997 (testing); 5000 random training samples for KRLS
- Implemented in Python using NumPy/SciPy and `LINPACK`'s "`dchud`" for rank-1 Cholesky update
- Hyperparameter optimization using grid search, $\lambda, \gamma \in \left\{2^{-20}, 2^{-18}, \ldots, 2^{18}, 2^{20}\right\}$
- 500 Random Features, averages over 25 runs

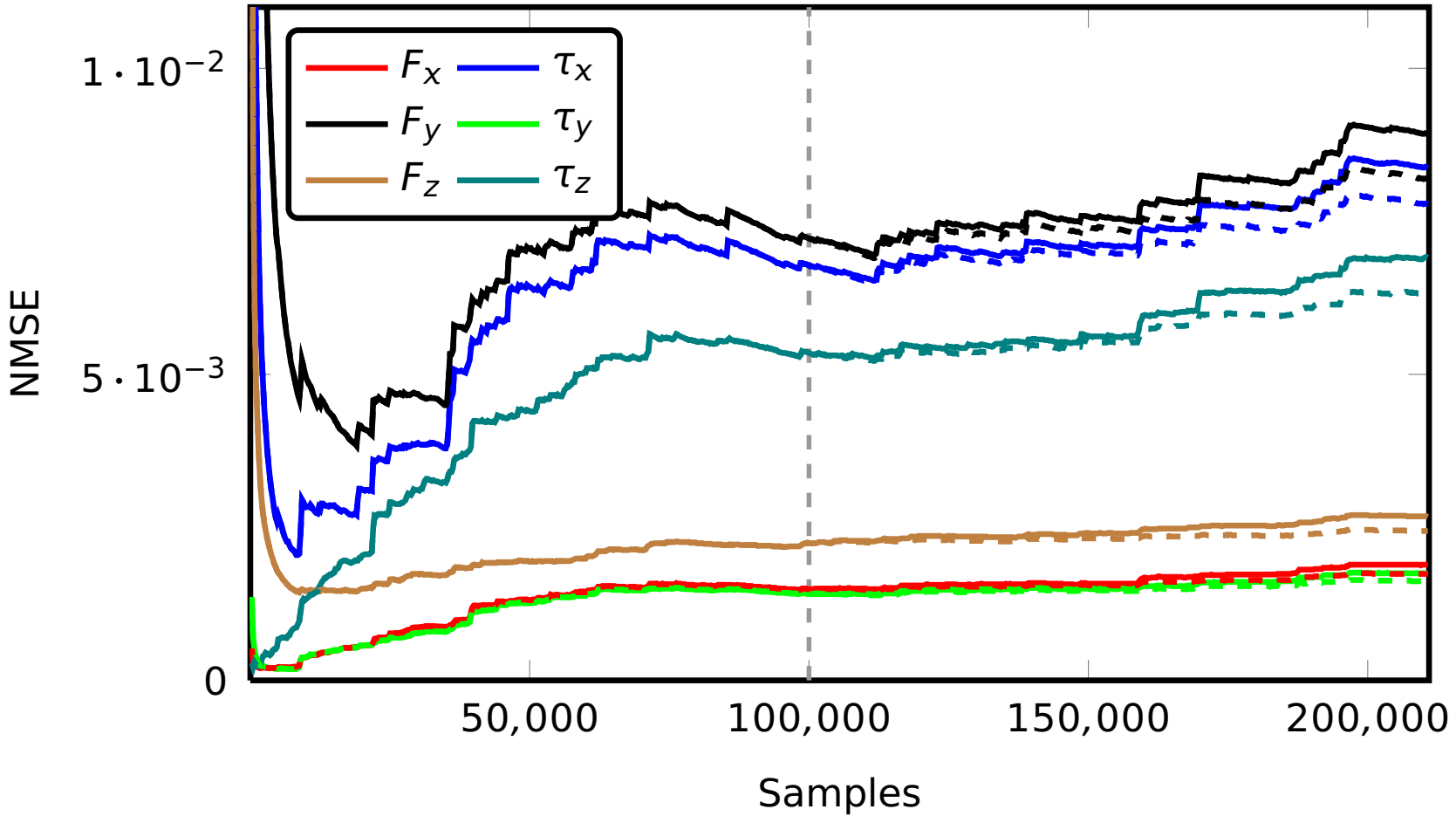
Generalization Performance

	KRLS	RLS+RF	(batch)
F_x	0.20	0.22 ± 0.03	...
F_y	0.93	1.00 ± 0.10	...
F_z	0.36	0.30 ± 0.01	...
τ_x	0.80	0.95 ± 0.10	...
τ_y	0.17	0.20 ± 0.03	...
τ_z	0.74	0.79 ± 0.05	...
T_{train}	$13.5s$	$1.4ms$ per sample	$10.0s$
$T_{predict}$	$37.0s$	$76\mu s$ per sample	$6.4s$

Online Prediction



Online Prediction



Ongoing Work

- Move to iCub, more data, more methods
- Prediction confidence to distinguish between prediction errors and external forces ($F_E = F - F_I$)
- Hybrid analytical/ML model; approach just published :((Nguyen-Tuong & Peters, 2010)
- “Proper” online methods with online-to-batch conversion ($\mathcal{O}(D^2) \rightarrow \mathcal{O}(D)$)

Ongoing: Prediction Confidence (1)

External forces on the arm can be obtained using $F_E = F - F_I$, where F_I is predicted by our inverse dynamics model and F is given by the current F/T sensor measurement.

But how can we distinguish between an external force F_E and prediction errors?

Solution: We need a measure of prediction confidence.

Ongoing: Prediction Confidence (2)

Using the standard linear Gaussian Process model (Rasmussen & Williams, 2005) instead of RLS, we obtain a *prediction distribution*

$$p(f|\mathbf{x}, \mathbf{X}, \mathbf{y}) = \mathcal{N}\left(\sigma_n^{-2}\mathbf{x}^\mathsf{T}\mathbf{A}^{-1}\mathbf{X}\mathbf{y},\ \mathbf{x}^\mathsf{T}\mathbf{A}^{-1}\mathbf{x}\right) \ ,$$

where $\mathbf{A} = \sigma_n^{-2}\mathbf{X}\mathbf{X}^\mathsf{T} + \mathbf{\Sigma}_p^{-1}$, σ_n^2 is the variance of the Gaussian output noise, and $\mathbf{\Sigma}_p$ is the covariance matrix of the Gaussian prior of $\mathbf{w}$. Note the equivalence with RLS if $\sigma_n = 1$ and $\mathbf{\Sigma}_p = \lambda^{-1}\mathbf{I}$.

Ongoing: Domain Knowledge

Linearity in the Parameters Property for rigid body dynamics (Spong *et al.*, 2005)

$$D(\mathbf{q}) + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q}) = R(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\mathbf{P} = \tau \ .$$

We can thus create a hybrid analytical/RF approach by constructing a mapping

$$\phi(\mathbf{x}) = \left[R(\mathbf{x})^{\mathsf{T}}, z_1(\mathbf{x}), \ldots, z_D(\mathbf{x})\right]^{\mathsf{T}} \quad \text{with } \mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}]$$

Similar approach published recently :(

(Nguyen-Tuong & Peters, 2010)

Conclusions

- Kernel Methods perform well on the inverse dynamics problem
- Combining Random Features with RLS, we can learn the inverse dynamics model incrementally
- # Random Features D can be used to balance between computational demand and accuracy
- $\mathcal{O}(1)$ time and space complexity guarantees that we can continue updating forever
- Could be applied to other robotics problems: kinematics, sensor prediction, etc.

Questions

References

Fumagalli, Matteo, Gijsberts, Arjan, Ivaldi, Serena, Jamone, Lorenzo, Metta, Giorgio, Natale, Lorenzo, Nori, Francesco, & Sandini, Giulio. 2010. Learning to Exploit Proximal Force Sensing: A Comparison Approach. *Pages 149–167 of: From Motor Learning to Interaction Learning in Robots*.

Nguyen-Tuong, Duy, & Peters, Jan. 2010 (05). Using Model Knowledge for Learning Inverse Dynamics. *In: 2010 IEEE International Conference on Robotics and Automation*.

Rahimi, Ali, & Recht, Benjamin. 2008. Random Features for Large-Scale Kernel Machines. *Pages 1177–1184 of:* Platt, J.C., Koller, D., Singer, Y., & Roweis, S. (eds), *Advances in Neural Information Processing Systems 20*. Cambridge, MA: MIT Press.

Rasmussen, Carl Edward, & Williams, Christopher K. I. 2005. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Spong, Mark W., Hutchinson, Seth, & Vidyasagar, Mathukumalli. 2005. *Robot Modelling and Control*. John Wiley & Sons, New York, USA.