Learning State-Dependent Losses for Inverse Dynamics Learning

Neha Das*, Kristen Morse*, Yixin Lin, Austin Wang, Akshara Rai, Franziska Meier

***Abstract*—Being able to quickly adapt to changes in dynamics is crucial in model-based control for object manipulation tasks. Specifically, data efficiency is key when aiming for fast adaptation of the inverse dynamics model parameters. The loss function is an important element in how effectively an optimizer updates model parameters. Thus, we propose to *learn how to adapt fast* by learning a loss function. More concretely, we propose to learn structured, state-dependent loss functions through a *meta-training* phase. We then replace standard losses with our learned losses during online adaptation tasks. We evaluate our proposed approach on inverse dynamics learning tasks on real hardware data. We find that structured learned losses improve online adaptation speed, when compared to standard loss functions.**

I. Introduction

Truly autonomous robots need to be able to adapt their internal models when interacting with the environment. For instance, a robot that has learned its own dynamics, needs to be able to adapt to the changes in dynamics caused by picking and placing a heavy object (Figure 1). Humans are remarkably good at adapting to such changes very quickly. And studies on how humans adapt to external forces suggest that we *learn how to adapt* to such changes in dynamics, and that we remember the adaptation strategy when encountering previously experienced perturbations [8].

The larger goal of our work is to learn representations that endow robots with similar capabilities. In this work, we specifically look at fast learning or adaptation of inverse dynamics for model-based control using structured learned loss. While many robots have analytical inverse dynamics models, they are often crude approximations to the actual dynamics of the robot. Furthermore, even if the models would be accurate enough for the robot's dynamics, they do not model the changes in dynamics caused by interaction with the environment. Thus, a lot of research has gone into data-driven methods for model-based control [16], with the hope that learning-based methods can either learn better models altogether [17, 19], or learn error-models on top of existing analytical ones [18].

In that context, online learning or adaptation of inverse dynamics models has been studied [21, 14, 6]. When such online learning methods are deployed on real robots, one has to be concerned with computational and data efficiency of the optimizer. An online learner needs to make the most out of the recently observed data with as few parameter updates as possible. A key insight of our work is that standard losses, such as the Mean-Squared-Error (MSE), are not state-dependent, but observed errors and how much we should correct for them can be state-dependent. For instance, an optimizer could update dynamics parameters aggressively throughout most of the state-space, but needs to be more conservative around singular configurations, or after overcoming the effects of static friction. Learned, state-dependent losses could potentially incorporate such issues and thus lead to faster inverse dynamics model learning. By adding this type of structure, we also are able to visualize more about our loss, allowing us some insight into the "black box" and consequently reducing the amount of researcher time used tuning our learning. Thus, in this work, we aim at improving data efficiency, by replacing standard loss functions for inverse dynamics learning, with learned, state-dependent losses.

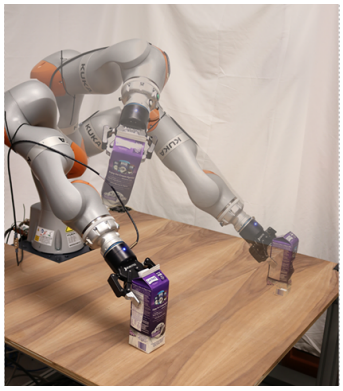


Fig. 1: Picking and placing a milk carton with KUKA.

Towards this goal, we develop a meta-learning framework, that trains loss functions for fast adaptation of inverse dynamics models. Our approach builds on [3], a gradient-based meta-learning framework for learning loss functions. This work proposed learning losses that are parameterized as neural networks for various learning tasks. Here, we contribute the following: 1) we utilize [3] to formulate an offline meta-training algorithm for training loss functions for inverse dynamics learning, and 2) we propose two structured loss representations specifically suitable for learning inverse dynamics models, one of which is state-dependent.

We evaluate our proposed loss representations on several inverse dynamics learning tasks on real hardware data of a 7 degree of freedom robot arm. Our results show that meta-learning structured losses leads to more robust meta-training of the losses, as compared to using typical neural network representations for the loss function. Furthermore, we find that our structured losses lead to faster learning and online adaptation of inverse dynamics models. Finally, we show that the proposed state-dependent loss outperforms all other loss representations in terms of adaptation speed, providing experimental validation for the thought that learning state-dependent loss functions can improve data efficiency in inverse dynamics learning.

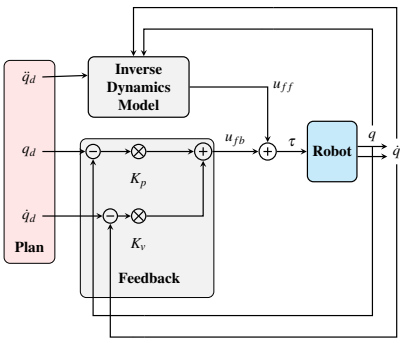


Fig. 2: The Inverse Dynamics Controller. See Section II for details

II. Background and Related Work

A typical model-based control loop for inverse dynamics control is shown in Figure 2. The controller accepts a target position q_{target} and outputs a plan - the desired joints, joint velocities and joint accelerations $Q_d = \{q_d, \dot{q}_d, \ddot{q}_d\}$ for each time step of the trajectory. The inverse dynamics model takes the current joint position q and velocity $\dot{q}$, as well as the desired acceleration $\ddot{q}_d$ as input and outputs a feed forward torque u_{ff}. The controller then combines u_{ff} with the feedback u_{fb} to produce the total desired torque τ applied on the robot. The feedback component is computed through a PD-control law that corrects for the errors caused due to an inaccurate dynamics model: $u_{fb} = K_p(q_d - q) + K_d(\dot{q}_d - \dot{q})$. u_{fb} corrects for inaccuracies in the dynamics model, and pushes the robot towards the desired trajectory. Ideally, for compliant and fast motion, we want small contributions from the feedback, and most of the torque to come from u_{ff}, motivating the need for learning accurate inverse dynamics models.

A. Learning Inverse dynamics

There has been significant work on computationally efficient methods for learning and adapting inverse dynamics models [9, 23, 20, 17, 6], including learning *incrementally* in an online manner [21, 4, 13, 11]. Other works assume a prior model which may be inaccurate in the real world, and attempts to learn a model over this error instead of the entire inverse dynamics model [18, 14, 15]. In contrast to our work, all of these approaches learn a dynamics model that optimizes the mean-squared error, and do not benefit from a state-dependent loss function that can lead to faster learning online.

B. Meta-Learning

Learning-to-adapt can be considered a form of meta-learning, where an agent attempts to learn *how to learn* from observations. We can segregate such approaches roughly into three classes: Methods that meta-learn initialization of models from which an agent can quickly adapt to new scenarios [5], methods that learn an optimizer representation [1, 12], and finally methods that learn loss functions [3, 10, 24], which we focus on. Specifically, this work builds on the work by [3]. We modify their framework for training structured loss functions for inverse dynamics learning, and use it to efficiently update dynamics models at test time.

III. Learning loss functions for Inverse Dynamics Models

Our loss-learning framework comprises of two phases: *Phase 1 (meta training):* A process by which we use data collected from a motion task to optimize the parameters ϕ of our loss function $\mathscr{L}_{\text{learn}_\phi}$. *Phase 2 (meta testing):* A process by which we train our inverse dynamics model by optimizing the $\mathscr{L}_{\text{learn}_\phi}$ with optimal parameters $\phi*$ to train f_θ on new tasks. We start out by recapping how to learn inverse dynamics models given any loss function.

A. Learning Models for Inverse Dynamics Control

We aim to learn a model of the inverse dynamics - i.e given the current joint positions q and velocities $\dot{q}$, and the next desired joint acceleration $\ddot{q}_d$, we want to learn a function f, parameterized by θ such that:

$$u_{ff} = f_\theta(q, \dot{q}, \ddot{q}_d)$$

where u_{ff} is the feed-forward torque that should be applied to achieve the desired acceleration $\ddot{q}_d$ as per the inverse dynamics model. In this work, we model f via a neural network parameterized by θ, and we aim to learn it from observed data as quickly as possible. This model is trained on a data set of N trajectories $\mathscr{D} = \{\{Q_t = (q_t, \dot{q}_t, \ddot{q}_{t+1}), \tau_t\}_{t=1}^{T}\}_{n=1}^{N}$ of length T by minimizing a loss $\mathscr{L}_{\text{MSE}}$ between the predicted control outputs (u_{ff}) and ground truth torque values (τ).

This loss is domain-independent and can be used for any function approximation problem. We believe that learning loss functions for inverse dynamics learning can lead to losses that update these models more effectively. Next, we propose loss functions that can be learned offline during a *meta-training* phase, and can then be used instead of the $\mathscr{L}_{\text{MSE}}$ to update the dynamics parameters θ.

B. Loss Function Representations

We explore the following loss architectures in our work:

1) Multi-Layer Perceptron (mlp): In [3] the learned loss is represented as a neural network. Similar to the MSE loss, the learned loss $l = \mathscr{L}_{\text{learn}_\phi}(\hat{\tau}, \tau)$ takes as input the predicted output $\hat{\tau} = f_\theta$ and its corresponding ground truth value τ, and outputs a loss value that computes an effective distance between f_θ and τ. The loss architecture is a fully-connected MLP network with 3 layers of 40 neurons each, with a ReLu activation in the hidden layer and a Softplus activation applied to the output. This loss representation is our baseline.

2) Structured loss (structured): The *mlp* loss architecture is a one size fits all solution that is not tailored to the structure of the inverse dynamics learning problem. Here, we propose to use a parameterized loss of the form:

$$\mathscr{L}_{\text{learn}_\phi} = \sum_{j=1}^{J} \phi_j (f_{\theta j} - \tau_j)^2, \tag{1}$$

where J denotes the number of controllable joints and $\phi_j \in \mathrm{R}^1$ are learn-able parameters, one per joint j. This structured learned loss takes into account to the shape of the inverse-dynamics model's outputs (i.e a predicted torque f_θ per joint),

and learns to assign different weights to the different joints. In general, dynamics errors in some joints can cause poor tracking of desired joint acceleration. Intuitively, this loss learns to weigh torque error on crucial joints more heavily than others. Updating more important joints faster likely leads to improvement in online learning speed.

3) State Dependent loss (state-dependent): Our proposed structured loss assumes constant learned weights for the whole robot state space. Here, we introduce a state-dependent loss with weights ϕ_j that are a function of the current joint state $\{q, \dot{q}\}$:

$$\mathscr{L}_{\text{learn}_\phi} = \sum_{j=1}^{J} \phi_j(q, \dot{q})(f_{\theta_j} - \tau_j)^2. \tag{2}$$

where the functions ϕ_j are represented as neural networks with hidden layer of size 32 with ReLUs as activation functions. State-dependent loss can improve upon structured loss by adapting to (ie reducing weight) areas of the state space where the robot has lower control authority (eg. around singularities)

Moreover, some state-action-next-state transitions are non-deterministic on hardware, due to effects like hysteresis. In such cases, a state-dependent loss function can learn to reduce the weights on torque errors in hard to approximate regions.

Algorithm 1 Learning Loss functions at (*meta-train*)

Randomly initialize ϕ
for each *epoch* **do**
  for each *batch* in *epoch* **do**
    Randomly initialize θ
    Sample $batch_A = Q_A, \tau_A$
    Sample $batch_B = Q_B, \tau_B$
    for each *iter* in $iters_{max}$ **do**
      Calculate θ_{new} *from* $\mathscr{L}_{learn_\phi}$ *with the old* ϕ
        $\theta_{new} \leftarrow \theta - \alpha.\nabla_\theta \mathscr{L}_{\text{learn}_\phi}(f_\theta(Q_A), \tau_A)$
      Update ϕ *based on* $f_{\theta_{new}}$*'s performance*
        $\phi \leftarrow \phi - \eta.\nabla_\phi \mathscr{L}_{\text{MSE}}(f_{\theta_{new}}(Q_B), \tau_B)$
    end for
  end for
end for

C. Learning loss functions

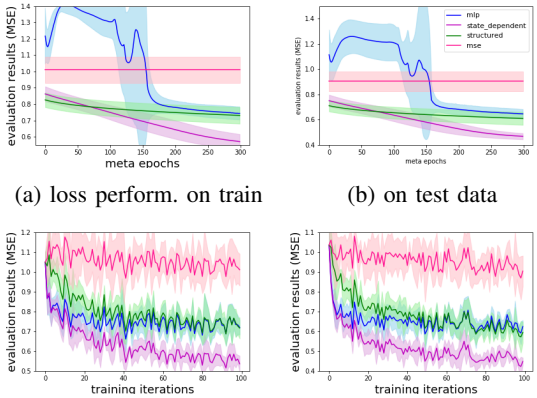
These loss functions can be learned offline, during a meta-training phase. We employ stochastic gradient descent (SGD) to update the parameters ϕ of the learned loss. Note that at each step of the optimization process, ϕ must be updated in a way such that training the model f_θ with the corresponding $\mathscr{L}_{\text{learn}_\phi}$ brings us a step closer to obtaining an accurate model of the robot's inverse-dynamics. This can be checked by evaluating the resultant inverse-dynamics model with $\mathscr{L}_{\text{MSE}}$.

More concretely, we first update the parameters θ via the update rule $\theta_{new} \longleftarrow \theta - \alpha\nabla_\theta \mathscr{L}_{\text{learn}_\phi}$ using batches $\{Q, \tau\}$ from the meta-train data set. We can now evaluate this updated model $f_{\theta_{new}}$, and therefore implicitly the $\mathscr{L}_{\text{learn}_\phi}$.

As per [3], we use the notion of a task loss to perform this evaluation. In our experiments, this is done using the Mean Squared Error function (formalized as $\mathscr{L}_{\text{MSE}}(f_{\theta_{new}}(Q), \tau)$). We can now use this metric to calculate an update to ϕ. In our SGD setting, we can write down the update rule for ϕ as:

$$\phi \longleftarrow \phi - \eta.\nabla_\phi \mathscr{L}_{\text{MSE}}(f_{\theta_{new}}(Q), \tau) \tag{3}$$

While it is possible to analytically derive this gradient (See [3]), one can use modern libraries for such meta-gradient computations. In particular, we use *higher* [7]. Our approach for meta-training a learned loss for learning inverse-dynamics models is summarized in Algorithm 1



(a) loss perform. on train (b) on test data

(c) optim. trace on train (d) on test data

Fig. 3: **Hardware meta-training phase:** (a) and (b) plot the performance of the learned losses as a function of meta-training epochs. (c) and (d) show loss traces during parameter learning of f_θ when using the final meta (learned) and fixed (mse) losses. The state-dependent loss is visibly more stable to train and is faster to optimize the inverse-dynamics model, compared to other learned losses. Results are averaged across 5 seeds.

IV. Hardware Experiments

We now turn to evaluating our proposed loss architectures on real hardware data.

A. Meta-training: Learning loss functions

We first evaluate the meta-training phase - in particular, how well our learned losses can optimize inverse dynamics models on unseen data. For these experiments, we collect sine motion data at frequencies $[0.02, 0.03, 0.04, 0.06, 0.08]$ for meta-training and $[0.05, 0.07]$ for meta-testing. Our controller runs at 250 Hz, and for each frequency we collect 30s of data.

Figures 3 (a) and (b) demonstrate that meta-training learned losses with a structure, (i.e, both *structured* and *state-dependent* losses) is much more stable than meta-training the *mlp* learned loss. From Figures 3(c) and (d), we again note that inverse-dynamics models train much faster when learned-losses are used for optimization as opposed to MSE loss in the order: *state-dependent* $>$ *structured* $>$ *mlp*.

B. Online Adaptation on Pick and Place Task

We now use the learned losses to online learn the inverse dynamics model of a pick and place task.

We collect 3 trials of picking-and-placing a milk carton, which weighs 857 grams, and separate the task into 5 motions: reaching for the carton (*reach*), lifting the carton (*lift*), moving the carton across the table (*move-over*), putting the carton

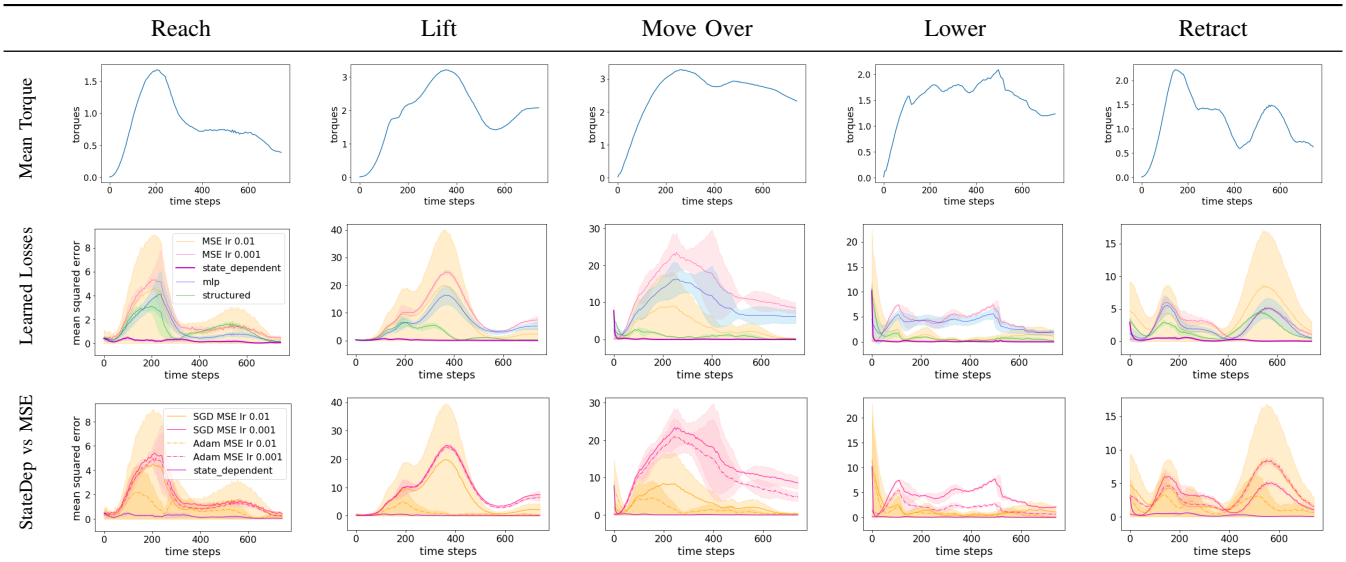


Fig. 4: **Online Adaptation on 5 motions**: The 1st row depicts a the mean torque across all joints while executing these motions. 2nd row depicts the performance (MSE) of f_θ as it is being adapted on the 5 motions using $\mathscr{L}_{\text{learn}_\phi}$ or $\mathscr{L}_{\text{MSE}}$. Models trained using *state-dependent* loss significantly outperforms all other losses including models trained using MSE losses with adaptive optimizers such as Adam (row 3)

down (*move-down*), and retracting the arm to it's rest posture (*retract*).

For each trial, the inverse-dynamics model f_θ is initialized randomly and then trained by online updating its parameters θ using the various learned losses on the stream of sequential data points $\{q_t, \dot{q}_t, \ddot{q}_{t+1}, \tau_t\}$ we receive while undergoing each of the above 5 motions. We train our model f_θ on each motion sequentially, always warm-starting the model for the next motion.

We record the MSE between predicted and actual torques while training on each task in sequence with the learned losses *mlp*, *structured* and *state-dependent* each with learning rates 0.001, as well as the MSE loss with learning rates 0.001 and 0.01 and plot these in Figure 4. The second row of the same Figure shows the training curves that use the learned losses on real robot sine-motion data. We observe that all learned losses outperform the standard MSE loss; most strikingly the state-dependent loss significantly outperforms all other losses. It is able to achieve low error predictions throughout all motions.

We note that models optimized with learned losses perform consistently better than those optimized using the MSE loss at the same learning rate - 0.001. We further note that the state-dependent loss outperforms all other losses (including the model trained with MSE loss at a higher learning rate, as well as an adaptive optimiser such as Adam).

This strongly supports our hypothesis that the loss landscape for inverse-dynamics is dependent on the current joint state $(q, \dot{q})$. Additionally, we observe that the evaluation curves for the state-dependent loss are much smoother and show less variance across multiple trajectories (which follow the same motion as we progress through training f_θ). This indicates that training a model with state-dependent loss on an online stream of data is significantly more stable than training with other learned-losses or MSE.

The variance of the other curves likely correlates to the rising and falling torque values throughout the motion. Most of the other models do not appear to adapt well to this variance, resulting in large errors.

The *state-dependent* loss is however, able to consistently produce low MSE metrics: which provides evidence that it adjusts well to unexpected changes in torque. Consequently, these results suggest that keeping track of state allows our learned loss to better adapt where torque values may be higher (for example, in cases of stiction in our robot joints).

V. Conclusion

In this work, we contribute a framework for learning loss functions, apply it to inverse dynamics models and improve upon previous loss-learning [3] methods by adding structure and state dependency to the loss representations. Our results show show that adding structure to the learned loss produces faster, better and more stable adaptation on hardware data when compared to learned losses without structure or standard domain-independent losses.

We see several promising directions for further research involving the methods we have introduced in this work. Currently, due to safety concerns, the online adaptation experiments on the hardware were performed using data collected offline from the robot, but fed to the model sequentially, mimicking an actual online setting. In future, we plan on investigating methods to safely apply meta-learning directly on robots. Another direction would be to investigate the *state-dependent* loss further in order to verify some of our hypotheses relating to its exceptional performance. For instance, we would like to visualize the inner workings of this loss. Given our proposed methods' suitability to online adaptation of any type of model, another interesting extension of this work could be adapting it to do parameter estimation for robots [2, 22] online.

References

[1] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances In Neural Information Processing Systems*, pages 3981–3989, 2016.

[2] Christopher G. Atkeson, Chae H. An, and John M. Hollerbach. Estimation of inertial parameters of manipulator loads and links. *The International Journal of Robotics Research*, 5(3):101–119, 1986. doi: 10.1177/027836498600500306.

[3] Sarah Bechtle, Artem Molchanov, Yevgen Chebotar, Edward Grefenstette, Ludovic Righetti, Gaurav Sukhatme, and Franziska Meier. Meta-learning via learned loss. *arXiv preprint arXiv:1906.05374*, 2019.

[4] Raffaello Camoriano, Silvio Traversaro, Lorenzo Rosasco, Giorgio Metta, and Francesco Nori. Incremental semiparametric inverse dynamics learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 544–550. IEEE, 2016.

[5] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/finn17a.html.

[6] Arjan Gijsberts and Giorgio Metta. Real-time model learning using incremental sparse spectrum gaussian process regression. *Neural networks*, 41:59–69, 2013.

[7] Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*, 2019.

[8] David J Herzfeld, Pavan A Vaswani, Mollie K Marko, and Reza Shadmehr. A memory of errors in sensorimotor learning. *Science*, 2014.

[9] Kevin Hitzler, Franziska Meier, Stefan Schaal, and Tamim Asfour. Learning and adaptation of inverse dynamics models: A comparison.

[10] Rein Houthooft, Yuhua Chen, Phillip Isola, Bradly Stadie, Filip Wolski, OpenAI Jonathan Ho, and Pieter Abbeel. Evolved policy gradients. In *Advances in Neural Information Processing Systems*, pages 5400–5409, 2018.

[11] Lorenzo Jamone, Bruno Damas, and José Santos-Victor. Incremental learning of context-dependent dynamic internal models for robot control. In *2014 IEEE international symposium on intelligent control (ISIC)*, pages 1336–1341. IEEE, 2014.

[12] Ke Li and Jitendra Malik. Learning to optimize. *arXiv preprint arXiv:1606.01885*, 2016.

[13] Franziska Meier, Philipp Hennig, and Stefan Schaal. Incremental local gaussian regression. In *Advances in Neural Information Processing Systems*, pages 972–980, 2014.

[14] Franziska Meier, Daniel Kappler, Nathan Ratliff, and Stefan Schaal. Towards robust online inverse dynamics learning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4034–4039. IEEE, 2016.

[15] Duy Nguyen-Tuong and Jan Peters. Using model knowledge for learning inverse dynamics. In *2010 IEEE international conference on robotics and automation*, pages 2677–2682. IEEE, 2010.

[16] Duy Nguyen-Tuong and Jan Peters. Model learning for robot control: a survey. *Cognitive processing*, 12(4):319–340, 2011.

[17] Duy Nguyen-Tuong, Jan R Peters, and Matthias Seeger. Local gaussian process regression for real time online model learning. In *Advances in neural information processing systems*, pages 1193–1200, 2009.

[18] Nathan Ratliff, Franziska Meier, Daniel Kappler, and Stefan Schaal. Doomed: Direct online optimization of modeling errors in dynamics. *Big data*, 4(4):253–268, 2016.

[19] Stefan Schaal and Christopher G Atkeson. Constructive incremental learning from only local information. *Neural computation*, 10(8), 1998.

[20] Stefan Schaal, Christopher G Atkeson, and Sethu Vijayakumar. Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence*, 17(1):49–60, 2002.

[21] Olivier Sigaud, Camille Salaün, and Vincent Padois. On-line regression algorithms for learning mechanical models of robots: a survey. *Robotics and Autonomous Systems*, 59(12):1115–1129, 2011.

[22] Giovanni Sutanto, Austin Wang, Yixin Lin, Mustafa Mukadam, Gaurav Sukhatme, Akshara Rai, and Franziska Meier. Encoding physical constraints in differentiable newton-euler algorithm. *arXiv preprint arXiv:2001.08861*, 2020.

[23] Sethu Vijayakumar and Stefan Schaal. Local dimensionality reduction for locally weighted learning. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.*, pages 220–225. IEEE, 1997.

[24] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2), 2002.