Proceedings of Machine Learning Research vol 120:1–10, 2020

Robust Online Model Adaptation by Extended Kalman Filter with Exponential Moving Average and Dynamic Multi-Epoch Strategy

Abulikemu Abuduweili ABDUWALI@PKU.EDU.CN
School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, P. R. China

Changliu Liu CLIU6@ANDREW.CMU.EDU
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract

High fidelity behavior prediction of intelligent agents is critical in many applications. However, the prediction model trained on the training set may not generalize to the testing set due to domain shift and time variance. The challenge motivates the adoption of online adaptation algorithms to update prediction models in real-time to improve the prediction performance. Inspired by Extended Kalman Filter (EKF), this paper introduces a series of online adaptation methods, which are applicable to neural network-based models. A base adaptation algorithm Modified EKF with forgetting factor (MEKF$_\lambda$) is introduced first, followed by exponential moving average filtering techniques. Then this paper introduces a dynamic multi-epoch update strategy to effectively utilize samples received in real time. With all these extensions, we propose a robust online adaptation algorithm: MEKF with Exponential Moving Average and Dynamic Multi-Epoch strategy (MEKF$_{\text{EMA-DME}}$). The proposed algorithm outperforms existing methods as demonstrated in experiments. The source code is open-sourced in the following link https://github.com/intelligent-control-lab/MEKF_MAME.

Keywords: Online adaptation, extended Kalman filter, exponential moving average, optimization

1. Introduction

Supervised learning has been widely used to obtain models to predict the behaviors of intelligent agents Rudenko et al. (2019). **Behavior prediction** is a sub-topic of time series prediction Weigend (2018), which includes but is not limited to vehicle trajectory prediction during autonomous driving Lefèvre et al. (2014) and human-motion prediction during human-robot collaboration Cheng et al. (2019). Although a trained model typically performs well on the training set, performance can drop significantly in a slightly different test domain or under a slightly different data distribution Si et al. (2019); Callison-Burch et al. (2010). For tasks without annotated corpora from the test domain, adaptation techniques are required to deal with the lack of domain-specific data. This paper studies robust online adaptation algorithms for behavior prediction.

In **online adaptation**, a prediction model observes instances sequentially over time. After every observation, the model outputs a prediction and receives the ground truth. Then the online adaptation algorithm updates the prediction model according to the error measured between the prediction and the ground truth. The goal of adaptation is to improve the prediction accuracy in subsequent rounds. An online adaptation algorithm is robust if it can efficiently adapt an existing model to a different (test) data distribution, without generating big transient errors.

For prediction models encoded in neural networks, most existing online adaptation approaches are based on stochastic gradients Kivinen et al. (2004). For example, the identification-based ap-

proach uses stochastic gradient descent (SGD) to adapt the model online Bhasin et al. (2012). However, these methods may be sub-optimal in minimizing the local prediction errors. Another solution is to use the recursive least square parameter adaptation algorithm (RLS-PAA) Ljung and Priouret (2010), which has been applied to adapt the last layer of a feedforward neural network Cheng et al. (2019) or the last layer of a recurrent neural network Si et al. (2019). RLS-PAA can only adapt the last layer of a neural network since it only applies to linear models. To adapt other layers, the adaptation problem becomes nonlinear, which requires the development of robust optimal nonlinear adaptation algorithms Jazwinski (2007); Cooper et al. (2014); Abuduweili et al. (2019).

Since a neural network parameterizes a nonlinear system with a layered structure, learning or adaptation of the neural network is equivalent to parameter estimation of the nonlinear system. The extended Kalman filter (EKF) is one promising method for nonlinear parameter estimation Jazwinski (2007), which is derived by linearizing the system equations at each time step and applying Kalman filter (an optimal filter that minimizes the tracking error) on the linearized system. The EKF approach has been demonstrated to be superior to the SGD-based algorithms in training feedforward neural networks Iiguni et al. (1992); Ruck et al. (1992). Nonetheless, in online adaptation, more recent data is more important Fink et al. (2001). Similar to adaptive EKF methods Yang et al. (2006); Ozbek and Efe (2004); Anderson and Moore (2012) that discount old measurements, this paper considers the Modified Extended Kalman Filter with forgetting factor, MEKF$_\lambda$, as a base adaptation algorithm.

On top of the base adaptation algorithm, the following modifications are made. Generally, the step size of parameter update in EKF-based approaches may not be optimal, due to the error introduced during linearization. Inspired by **exponential moving average (EMA)** methods, this paper proposes EMA filtering to the base MEKF$_\lambda$ in order to increase the convergence rate. The resulting algorithm is called MEKF$_{\text{EMA}}$. Then in order to effectively utilize the samples in online adaptation, this paper proposes a **dynamic multi-epoch update** strategy to discriminate the "hard" samples from "easy" samples, and sets different weights for them. The dynamic multi-epoch update strategy can improve the effectiveness of online adaptation with any base optimizers, e.g., SGD or MEKF$_{\text{EMA}}$. By incorporating MEKF$_{\text{EMA}}$ with the dynamic multi-epoch update strategy, we propose the algorithm MEKF$_{\text{EMA-DME}}$ (MEKF with Exponential Moving Average and Dynamic Multi-Epoch update strategy).

The remainder of the paper first formulates the online adaptation problem, then discusses the proposed algorithm, and finally validates the effectiveness and flexibility of the proposed algorithms.

2. Online adaptation framework

The behavior prediction problem is to make inference on the future behavior of the target agent given the past and current measurement of the target agent and its surrounding environment. The transition model for behavior prediction problem is formulated as

$$\mathbf{Y}_t = f(\boldsymbol{\theta}, \mathbf{X}_t), \tag{1}$$

where the input vector $\mathbf{X}_t = [\boldsymbol{x_t}; \boldsymbol{x_{t-1}}; \cdots; \boldsymbol{x_{t-n+1}}]$ denotes the stack of n-step current and past measurements (e.g. trajectory of states or extracted features) at time steps $t, t-1, \ldots, t-n+1$. The output vector $\mathbf{Y}_t = [\boldsymbol{y_{t+1}}; \boldsymbol{y_{t+2}}; \cdots; \boldsymbol{y_{t+m}}]$ denotes the stack of the m-step future behavior (e.g. future trajectory) at time steps $t+1, t+2, \ldots, t+m$. The function f is the prediction model that maps the measurements to the future behavior. $\boldsymbol{\theta}$ denotes the (ground truth) parameter of the model. It is assumed that there are recurrent structures in f such that the prediction of $\mathbf{Y}_t$ is made

by rolling out the one-step predictions $\boldsymbol{y_{t+1}} = f_1(\boldsymbol{\theta}, \mathbf{X}_t)$. The function f_1 is the one-step prediction function and is a recurrent part of the overall prediction model.

Online adaptation explores local overfitting to minimize the prediction error. At time step t, the following prediction error is to be minimized

$$\min_{\hat{\boldsymbol{\theta}}_t} \|\mathbf{Y}_t - f(\hat{\boldsymbol{\theta}}_t, \mathbf{X}_t)\|_p, \tag{2}$$

where $\mathbf{Y}_t$ is the ground truth trajectory (to be observed in the future) and $\hat{\mathbf{Y}}_t := f(\hat{\boldsymbol{\theta}}_t, \mathbf{X}_t)$ is the predicted trajectory using the estimated model parameter $\hat{\boldsymbol{\theta}}_t$. The adaptation objective can be in any ℓ_p norm. This paper considers ℓ_2 norm. Assume that the true model parameter changes slowly during adaptation, i.e., $\dot{\boldsymbol{\theta}} \approx 0$. Then the estimated model parameter that minimizes the prediction error in the future can be approximated by the estimated parameter that minimizes the fitting error in the past. Solving for the estimated parameter that minimizes the fitting error corresponds to a nonlinear least square (NLS) problem.

Definition 1 (Problem NLS) *Given a dataset* $\{(\mathbf{X}_i, \mathbf{Y}_i), i = 1, 2, \cdots, T\}$, *find* $\hat{\boldsymbol{\theta}}_t \in \mathbb{R}^n$ *that minimizes* $J_t(\hat{\boldsymbol{\theta}}_t) = \frac{1}{t}\sum_{i=1}^{t} \|\mathbf{e}_i\|_2^2$, *where error term is defined as* $\mathbf{e}_i = \mathbf{Y}_i - f(\hat{\boldsymbol{\theta}}_t, \mathbf{X}_i)$.

In online adaptation, the estimate of the model parameter is updated iteratively when new data is received. Then a new prediction is made using the new estimate. In the next time step, the estimate will be updated again given the new observation and the process repeats. It is worth noting that the observation we received at time t is $\boldsymbol{y_t}$. The other terms in $\mathbf{Y_t}$ remain unknown. This paper is focused on adaptation methods using only one-step observation. It is possible to adapt with multi-step observations, which will be studied in the future. The process for online adaptation is summarized in algorithm 1. $\hat{\boldsymbol{\theta}}_t$ is the estimate of the model parameter $\boldsymbol{\theta}$ at time t.

Algorithm 1 Generic Online Adaptation (Adaptable Prediction)

Input: Initial model parameters $\hat{\boldsymbol{\theta}}_\mathbf{0}$
Output: sequence of predictions $\{\hat{\mathbf{Y}}_t\}_{t=1}^T$
1: **for** $t = 1, 2, \cdots, T$ **do**
2: obtain the ground truth observation value $\boldsymbol{y_t}$; construct the input features $\mathbf{X}_t$
3: adaptation step (supervised): $\hat{\boldsymbol{\theta}}_t = \mathrm{Adapt}(\hat{\boldsymbol{\theta}}_{t-1}, \hat{\boldsymbol{y}}_\boldsymbol{t}, \boldsymbol{y_t})$
4: prediction step: $\hat{\mathbf{Y}}_t = f(\hat{\boldsymbol{\theta}}_t, \mathbf{X}_t)$, where $\hat{\mathbf{Y}}_t = [\hat{\boldsymbol{y}}_{\boldsymbol{t+1}}; \hat{\boldsymbol{y}}_{\boldsymbol{t+2}}; \cdots; \hat{\boldsymbol{y}}_{\boldsymbol{t+m}}]$
5: **end for**
6: **return** sequence of predictions $\{\hat{\mathbf{Y}}_t\}_{t=1}^T$

3. Robust nonlinear adaptation algorithms

3.1. Modified EKF with forgetting factor

Our base adaptation algorithm is inspired by the recursive EKF method Moriyama et al. (2003); Alessandri et al. (2007). In EKF, the object being estimated is the state value of a dynamic system, while in adaptable prediction, the object to be adapted is the parameters that describe the system dynamics. Nonetheless, we can apply the EKF approach to adapt model parameters by regarding model parameters as system states. By assuming that the ground truth $\boldsymbol{\theta}$ changes very slowly, we can pose the parameter adaptation problem as a static state estimation problem Ruck et al. (1992); Nelson (2000) with the following dynamics,

$$\hat{\boldsymbol{\theta}}_t = \hat{\boldsymbol{\theta}}_{t-1} + \boldsymbol{\omega}_t, \tag{3}$$

$$\boldsymbol{y_t} = f_1(\hat{\boldsymbol{\theta}}_{t-1}, \mathbf{X}_{t-1}) + \mathbf{u}_t, \tag{4}$$

where $\hat{\boldsymbol{\theta}}_t$ is an estimate of the (ground truth) model parameter $\boldsymbol{\theta}$; $\boldsymbol{y_t}$ is the observation at time t; and $\hat{\boldsymbol{y}}_{\boldsymbol{t}} = f_1(\hat{\boldsymbol{\theta}}_{t-1}, \mathbf{X}_{t-1})$ is the prediction for time step t made at time $t-1$. f_1 is the one-step prediction function. The injected process noise $\boldsymbol{\omega}_t \sim \mathcal{N}(0, \mathbf{Q}_t)$ and the injected measurement noise $\mathbf{u}_t \sim \mathcal{N}(0, \mathbf{R}_t)$ are assumed to be zero mean Gaussian white noise, and are identically and independently distributed. The symbol $\mathcal{N}$ represents Gaussian distribution. $\mathbf{Q}_t$ and $\mathbf{R}_t$ represent the covariance matrices for the process noise and the measurement noise respectively, which should be positive semidefinite. If there is no knowledge about the cross correlation of the noises, it is reasonable to assume that the entries in the noise vector are independent of each other, and set $\mathbf{Q}_t$ and $\mathbf{R}_t$ to be proportional to the identity matrix. For simplicity, we assume $\mathbf{Q}_t = \sigma_q \mathbf{I}$ and $\mathbf{R}_t = \sigma_r \mathbf{I}$ for $\sigma_q \geq 0$ and $\sigma_r > 0$.

In online adaptation, we assume that data in the distant past is no longer relevant for modeling the current dynamics, i.e. more recent data is more important. Hence, we consider a weighted nonlinear recursive least squares (NLS) problem:

$$\min_{\hat{\boldsymbol{\theta}}_t} \sum_{i=1}^{t} \lambda^{t-i} \|\boldsymbol{y_i} - f_1(\hat{\boldsymbol{\theta}}_{i-1}, \mathbf{X}_{i-1})\|_2^2, \qquad 0 < \lambda \leq 1, \tag{5}$$

where λ is the "forgetting factor" which provides exponential decay to older samples. The forgetting factor prevents the EKF from saturation, and increases the algorithm's ability to track a changing system. Algorithm 2 summarizes the modified extended Kalman filter algorithm with forgetting factor (**MEKF**$_\lambda$).

Algorithm 2 Modified EKF algorithm with forgetting factor (MEKF$_\lambda$)

Input: Initial hyper-parameter for MEKF$_\lambda$: $p_0 > 0,\ \lambda > 0,\ \sigma_r > 0,\ \sigma_q \geq 0;\ \mathbf{P}_0 = p_0\mathbf{I}$
Input: previous parameter $\hat{\boldsymbol{\theta}}_{t-1}$, previous prediction $\hat{\boldsymbol{y}}_{\boldsymbol{t}} = f_1(\hat{\boldsymbol{\theta}}_{t-1}, \mathbf{X}_{t-1})$, current observation $\boldsymbol{y_t}$ at time step t
Output: Adapted parameter $\hat{\boldsymbol{\theta}}_{\mathbf{t}}$

$\mathbf{H}_t = \frac{\partial f_1(\boldsymbol{\theta}, \mathbf{X})}{\partial \boldsymbol{\theta}} |_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_{t-1}, \mathbf{X}=\mathbf{X}_{t-1}}$
$\mathbf{K}_t = \mathbf{P}_{t-1} \cdot \mathbf{H}_t^\mathsf{T} \cdot (\mathbf{H}_t \cdot \mathbf{P}_{t-1} \cdot \mathbf{H}_t^\mathsf{T} + \sigma_r \mathbf{I})^{-1}$
$\hat{\boldsymbol{\theta}}_t = \hat{\boldsymbol{\theta}}_{t-1} + \mathbf{K}_t \cdot (\boldsymbol{y_t} - f_1(\hat{\boldsymbol{\theta}}_{t-1}, \mathbf{X}_{t-1}))$
$\mathbf{P}_t = \lambda^{-1}(\mathbf{P}_{t-1} - \mathbf{K}_t \cdot \mathbf{H}_t \cdot \mathbf{P}_{t-1} + \sigma_q \mathbf{I})$

In algorithm 2, $\mathbf{K}_t$ is the Kalman gain. $\mathbf{P}_t$ is a matrix representing the uncertainty in the estimates of the model parameter $\boldsymbol{\theta}$. $\mathbf{H}_t$ is the gradient matrix by linearizing the network. In online adaptation, $\boldsymbol{\theta}_0$ is initialized by the offline trained parameter of the model. For $\mathbf{P}_0$, due to the absence of a priori information, the $\mathbf{P}_0$ matrix can be set to be proportional to the identity matrix, i.e. $\mathbf{P}_0 = p_0\mathbf{I}$ for $p_0 > 0$.

3.2. Extensions with exponential moving average filtering

In the following discussion, for simplicity, an optimizer (e.g. MEKF$_\lambda$) that solves the adaptation problem will be denoted as $A_{\mathbf{P}}$ with internal state matrix $\mathbf{P}$. The optimization process for adaptation can be compactly written as

$$\begin{aligned} \hat{\boldsymbol{\theta}}_t &= \hat{\boldsymbol{\theta}}_{t-1} + \mathbf{V}_t, \\ \mathbf{V}_t &= A_{\mathbf{P}}(\hat{\boldsymbol{\theta}}_{t-1}, \hat{\boldsymbol{y}}_{\boldsymbol{t}}, \boldsymbol{y_t}). \end{aligned} \tag{6}$$

where $\mathbf{V}_t$ is the step size of the parameter update at time step t.

To speed up the convergence of parameter estimation, we propose to apply exponentially-decayed moving average (EMA) for filtering in the MEKF$_\lambda$ optimization process. In SGD-based methods, numerous variants of EMA have been successfully used to speed up the convergence, including Polyak averaging Polyak (1964) and momentum Qian (1999). For MEKF$_\lambda$, we can either apply EMA on the step size $\mathbf{V}$, to be discussed as EMA-V or momentum; or apply EMA on the optimizer's inner state $\mathbf{P}$, to be discussed as EMA-P.

EMA-V EMA-V or momentum is widely used in SGD-based optimization algorithms Qian (1999), which helps accelerate gradient-based optimizers in relevant directions and dampen oscillations Qian (1999). Momentum can be regarded as an EMA filter on the step size of parameter update. It calculates the step size $\mathbf{V}_t$ by decreasing exponentially the older step size with a factor $\mu_v \in [0, 1]$, i.e. $\mathbf{V}_t = \mu_v \mathbf{V}_{t-1} + (1 - \mu_v) A_{\mathbf{P}}(\hat{\boldsymbol{\theta}}_{t-1}, \hat{\boldsymbol{y}}_{\boldsymbol{t}}, \boldsymbol{y}_{\boldsymbol{t}})$.

EMA-P As mentioned earlier in MEKF$_\lambda$, $\mathbf{P}_t$ is a matrix representing the uncertainty in the parameter estimates. In order to attenuate instability during adaptation caused by anomaly data, we can smooth the inner state of the optimizer by pre-filtering $\mathbf{P}_t$. The principle of pre-filtering the inner state (e.g., gradient, adaptive learning rate) before using them in optimization is applicable to many optimization algorithms. For example, in Adam Kingma and Ba (2014), the estimate of the first and second moment is filtered every step using EMA. Similarly, we can apply EMA on the inner state matrix $\mathbf{P}_t$ of MEKF$_\lambda$.

By combining EMA-V and EMA-P, we propose the modified extended Kalman filter with exponential moving average (**MEKF**$_{\text{EMA}}$) algorithm as shown in algorithm 3, where μ_v is a momentum factor, and μ_p is a decay factor for the EMA filtering of $\mathbf{P}_t$.

Algorithm 3 Modified Extended Kalman Filter with Exponential Moving Average Filtering

Input: Initial base hyper-parameter: $p_0 > 0,\ \lambda > 0,\ \sigma_r > 0,\ \sigma_q \geq 0; \mathbf{P}_0 = p_0 \mathbf{I}$
Input: Initial EMA hyper-parameter: $0 \leq \mu_v < 1,\ 0 \leq \mu_p < 1$
Input: previous parameter $\hat{\boldsymbol{\theta}}_{t-1}$, previous prediction $\hat{\boldsymbol{y}}_{\boldsymbol{t}} = f_1(\hat{\boldsymbol{\theta}}_{t-1}, \mathbf{X}_{t-1})$, current observation $\boldsymbol{y}_{\boldsymbol{t}}$ at time step t
Output: Adapted parameter $\hat{\boldsymbol{\theta}}_t$

$\mathbf{H}_t = \frac{\partial f_1(\boldsymbol{\theta}, \mathbf{X})}{\partial \boldsymbol{\theta}} \mid_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_{t-1}, \mathbf{X}=\mathbf{X}_{t-1}}$
$\mathbf{K}_t = \mathbf{P}_{t-1} \cdot \mathbf{H}_t^\intercal \cdot (\mathbf{H}_t \cdot \mathbf{P}_{t-1} \cdot \mathbf{H}_t^\intercal + \sigma_r \mathbf{I})^{-1}$
$\mathbf{V}_t = \mu_v \mathbf{V}_{t-1} + (1 - \mu_v) \mathbf{K}_t \cdot (\boldsymbol{y}_{\boldsymbol{t}} - f_1(\hat{\boldsymbol{\theta}}_{t-1}, \mathbf{X}_{t-1}))$
$\hat{\boldsymbol{\theta}}_t = \hat{\boldsymbol{\theta}}_{t-1} + \mathbf{V}_t$
$\mathbf{P}_t^* = \lambda^{-1}(\mathbf{P}_{t-1} - \mathbf{K}_t \cdot \mathbf{H}_t \cdot \mathbf{P}_{t-1} + \sigma_q \mathbf{I})$
$\mathbf{P}_t = \mu_p \mathbf{P}_{t-1} + (1 - \mu_p) \mathbf{P}_t^*$

3.3. Dynamic multi-epoch update strategy

In generic online adaptation, all data are equally considered. We run the adaptation algorithm chronologically from the first data $\mathbf{X}_1$ to the last data $\mathbf{X}_T$. Every data sample is used only once, as shown in algorithm 1. We call the adaptation method that uses every data sample only once as **single-epoch online update strategy**.

Inspired by curriculum learning Bengio et al. (2009) in offline training, we introduce a more effective way to determine the adaptation epochs for every data sample during online adaptation. A curriculum can be viewed as a sequence of training criteria. Each training criterion in the sequence is associated with a different set of weights on the training examples. That said, it is practically

useful to differentiate "easy" samples from "hard" samples. In the online adaptation scenario, we introduce the following dynamic multi-epoch strategy to mimic curriculum learning.

Definition 2 (Dynamic multi-epoch online update strategy) *In online adaptation, the predicted output $\hat{\boldsymbol{y}}_t$ generated by the estimated parameter $\boldsymbol{\theta}^*$ is $\hat{\boldsymbol{y}}_t = f_1(\boldsymbol{\theta}^*, \mathbf{X}_{t-1})$. Define a criterion $\mathcal{C}$ to determine the number of epochs κ_t ($\kappa_t \in \mathbb{N}$) to adapt the parameter with the current sample, i.e., $\kappa_t = \mathcal{C}(\mathbf{X}_{t-1}, \boldsymbol{y}_t, \hat{\boldsymbol{y}}_t, \boldsymbol{\theta}^*)$. In other words, we reuse the input-output pair $(\mathbf{X}_{t-1}, \boldsymbol{y}_t)$ κ_t times to adapt the parameter $\boldsymbol{\theta}^*$. This approach is called the dynamic multi-epoch online update strategy or* ***dynamic multi-epoch update****.*

We propose a very straighforward criterion $\mathcal{C}$ to determine the number of epochs κ_t for every sample, as shown in algorithm 4[1]. Two thresholds ξ_1 and ξ_2 are used to discriminate "easy", "hard", and "anomaly" samples. Before updating the parameter, we calculate the prediction error $j_t = \|\boldsymbol{y}_t - \hat{\boldsymbol{y}}_t\|_2$ at the current step. If the error satisfies $j_t < \xi_1$, the sample is considered as an "easy" sample. Then we run single-epoch update for this sample. If the error satisfies $\xi_1 \leq j_t < \xi_2$, the sample is considered as a "hard" sample. Then we reuse this sample and run the adaptation twice. The rationale is that for a "hard" sample, an adaptation optimizer may not learn enough under single-epoch update. If the error satisfies $j_t \geq \xi_2$, the sample is considered as an "anomaly" sample. Then we skip the update of $\hat{\boldsymbol{\theta}}_t$. The rationale is that if the cost is too high, the sample is likely to be an anomaly point in the data distribution, which may destabilize the model adaptation process if learned. It is crucial to identify and learn more from those "hard" samples without sacrificing the generalizability of the model.

Algorithm 4 Dynamic Multi-epoch Update Strategy

Input: threshold $0 \leq \xi_1 \leq \xi_2$, optimizer $A_{\mathbf{P}}$ (e.g. $\mathrm{MEKF}_{\mathrm{EMA}}$), model prediction function f_1
Input: previous parameter $\hat{\boldsymbol{\theta}}_{t-1}$, previous input $\mathbf{X}_{t-1}$, previous prediction $\hat{\boldsymbol{y}}_t = f_1(\hat{\boldsymbol{\theta}}_{t-1}, \mathbf{X}_{t-1})$, current observation $\boldsymbol{y}_t$
Output: Adapted parameter $\hat{\boldsymbol{\theta}}_t$

$j_t = \|\boldsymbol{y}_t - \hat{\boldsymbol{y}}_t\|_2$
$\kappa_t = \begin{cases} 1\ , \textbf{ If } j_t < \xi_1 \\ 2\ , \textbf{ If } \xi_1 \leq j_t < \xi_2 \\ 0\ , \textbf{ If } j_t \geq \xi_2 \end{cases}$
$\boldsymbol{\theta}^*_{t,0} = \hat{\boldsymbol{\theta}}_{t-1}$
for $i = 1, \cdots, \kappa_t$ **do**
    $\hat{\boldsymbol{y}}^*_t = f_1(\boldsymbol{\theta}^*_{t,i-1}, \mathbf{X}_{t-1})$
    $\boldsymbol{\theta}^*_{t,i} = \boldsymbol{\theta}^*_{t,i-1} + A_{\mathbf{P}}(\boldsymbol{\theta}^*_{t,i-1}, \hat{\boldsymbol{y}}^*_t, \boldsymbol{y}_t)$
end for
$\hat{\boldsymbol{\theta}}_t = \boldsymbol{\theta}^*_{t,\kappa_t}$

The thresholds ξ_1 and ξ_2 can be determined by the validation set empirically. If the dataset is noise-free, there is no need to identify "anomaly" samples and we set $\xi_2 \to +\infty$. In general, we recommend the following method to find the desired ξ_1 and ξ_2. First, we need to run the single-epoch adaptation on the validation set and record each sample's prediction error $\{j_1, j_2, \cdots\}$. Second, we set ξ_1 as the $50\% \sim 95\%$ quantile value of the errors, and set ξ_2 as the $99.9\% \sim 100\%$ quantile value

1. The criterion can be application-specific. This criterion is proposed since it is observed in our experiment that a large number of epochs will lead to overfitting to the historical data, which is harmful to generalization. We will investigate more reasonable and effective criterion for dynamic multi-epoch update in the future.

of the errors. That means, we regard $50\% \sim 95\%$ of the samples as "easy" samples, $5\% \sim 50\%$ of the samples as "hard" samples, and $0\% \sim 0.1\%$ of the samples as "anomaly" samples.

We use $\mathbf{MEKF_{EMA\text{-}DME}}$ to denote MEKF_{EMA} with the dynamic multi-epoch update strategy.

4. Experiments

Experimental design In the experiments, we consider multi-task prediction problems for simultaneous intention and trajectory prediction of either humans or vehicles. We construct Recurrent Neural Network Salehinejad et al. (2017) (RNN) based architectures to conduct experiments on Mocap dataset (human) and NGSIM dataset (vehicle) Colyar and Halkias (2007). We leaved the details of the experiment in the appendix in the extended version of this paper Abuduweili and Liu (2019). Before online adaptation, the prediction models are trained offline. In the following discussion, we studied the performance of various adaptation algorithms on these offline-trained models (with online adaptation on the test set). In particular, we evaluate the accuracy (0-1) for intention prediction, and the mean squared error (MSE) for trajectory prediction.

Comparison among different optimizers The proposed algorithm $\text{MEKF}_{\text{EMA-DME}}$ is compared with the based algorithm MEKF_{λ} and other commonly used optimizers such as SGD, Adam, and Amsgrad. For fair comparison, we apply the dynamic multi-epoch update strategy on SGD, Adam, and Amsgrad.

Table 1: Comparison of different optimizers.

Dataset	Metrics	w/o adapt	SGD	Adam	Amsgrad	MEKF_{λ}	$\text{MEKF}_{\text{EMA-DME}}$
Mocap	accuracy	0.984	0.984	0.984	0.984	0.985	**0.985**
	MSE(dm^2)	3.271	3.185	3.149	3.156	2.788	**2.746**
NGSIM	accuracy	0.951	0.954 (.0427)	0.951 (.0426)	0.951 (.0427)	0.956 (.0430)	**0.956** (.0430)
	MSE(m^2)	2.559	2.367 (1.981)	2.402 (1.975)	2.407 (1.993)	2.157 (1.902)	**2.092** (1.893)

Table 1 shows the prediction performance of online adapted models using different optimizers on the Mocap dataset and the NGSIM dataset. Compared to the stochastic gradient-based algorithms, the EKF-based methods MEKF_{λ} and $\text{MEKF}_{\text{EMA-DME}}$ perform better. In addition, $\text{MEKF}_{\text{EMA-DME}}$ has the best performance among all, due to the extensions inspired by EMA and dynamic multi-epoch update. On the CMU Mocap dataset, Adam reduces the trajectory MSE by 3.73%. MEKF_{λ} reduces the trajectory MSE by 14.77%. $\text{MEKF}_{\text{EMA-DME}}$ reduces the trajectory MSE by 16.05%. These improvements are important to ensure safe and efficient operation of behavior prediction Zhao et al. (2020). The variance of perfomance using different optimizers on the NGSIM dataset is shown in the parenthesis. In particular, the standard deviation (Std.) of the prediction accuracy as well the Std. of MSE are shown. For intention prediction, MEKF_{λ} and $\text{MEKF}_{\text{EMA-DME}}$ have slightly higher Std. than other SGD based optimizers. However, for trajectory prediction, MEKF_{λ} and $\text{MEKF}_{\text{EMA-DME}}$ have lower Std. than other SGD based optimizers.

The running time of the adaptation algorithm correlates with the number of parameters to adapt. For the time complexity analysis, SGD, Adam, and Amsgrad have similar complexity, while MEKF_{λ} and MEKF_{EMA} have similar complexity. So we only compare MEKF_{λ} with SGD below. For adapting encoder's hidden layers (12480 parameters), SGD takes 0.08 seconds per sample and MEKF_{λ} takes 0.41 seconds per sample[2]. For real-time adaptable prediction, as long as the number of parameters to adapt is not too big, MEKF_{λ} can meet the real-time requirements.

2. The running time is evaluated on the NGSIM dataset using a Ubuntu desktop with Intel Core i9-9940X CPU (3.30GHz) and GeForce RTX 2080 Ti GPU.

Effectiveness of extensions This section studies the effectiveness of the proposed extensions to MEKF$_\lambda$ in sections 3.2 and 3.3. The hyperparameters are set as $\mu_p = 0.3$ and $\mu_v = 0.3$. The results are shown in table 2.

1. EMA-V or momentum barely improves the performance. Two potential reasons are: 1) The momentum does not help EKF-based optimizers. In every optimization step, EKF-based optimizers has already incorporated the historical data. Hence its step size $\mathbf{V}_t$ is already closer to optimum than that of SGD. The learning gain in SGD is not based on historical data but manually defined. 2) The moving average on the parameter or the step size is more applicable to offline training than to online adaptation. The inapplicability is due to the fact that online adaptation can only process data sequentially in time, which is significantly different from the shuffled, repetitive, and batched process in offline training.
2. EMA-P slightly improves the performance of MEKF$_\lambda$. Filtering of $\mathbf{P}_t$ can smooth the inner state and improve convergence.
3. Dynamic multi-epoch update improves the performance of MEKF$_\lambda$, and it has the best performance among all the proposed extensions.

We design the additional experiment about different criteria for DME in Abuduweili and Liu (2019) to compared the proposed criteria (Under the error spectrum, the first 50% are "easy" samples, the middle 50% to 99.9% are "hard" samples, and the last 0.1% are "anomaly" samples.) with fixed 2-epoch criteria (Each sample was used twice.) and random criteria (Which has same "easy" and "hard" ratio as the proposed criterion, but distinguishing "easy", "hard" and "anomaly" samples randomly.). The results in Abuduweili and Liu (2019) show that: the proposed criterion outperforms other criteria, which justifies the effectiveness of the proposed error-based criterion. Nonetheless, we will investigate more reasonable and effective criterion for dynamic multi-epoch update in the future.

Table 2: Performance of MEKF$_\lambda$ extensions.

Dataset	Metrics	MEKF$_\lambda$	MEKF$_\lambda$ + EMA-V	MEKF$_\lambda$ + EMA-P	MEKF$_\lambda$ + DME
Mocap	accuracy	0.985	0.985	0.985	0.985
	MSE(dm^2)	2.788	2.790	2.775	2.749
NGSIM	accuracy	0.956	0.956	0.956	0.956
	MSE(m^2)	2.157	2.156	2.123	2.122

5. Conclusions

This paper studied online adaptation of neural network-based prediction models for behavior prediction. An EKF-based adaptation algorithm MEKF$_\lambda$ was introduced as an effective base algorithm for online adaptation. In order to improve the performance and convergence of MEKF$_\lambda$, exponential moving average filtering was investigated, including momentum and EMA-P. Then this paper introduced a dynamic multi-epoch update strategy, which is compatible with any optimizer. By combining all extensions with the base MEKF$_\lambda$ algorithm, we introduced the robust online adaptation algorithm MEKF$_{\text{EMA-DME}}$. In the experiments, we demonstrated the effectiveness of the proposed adaptation algorithms.

In the future, mathematical analysis of the proposed online adaptation algorithm MEKF$_{\text{EMA-DME}}$ will be performed in order to provide theoretical guarantees on stability, convergence, and boundedness. In addition, we will apply the proposed algorithm on a wider range of problems in addition to behavior prediction problems.

References

Abulikemu Abuduweili and Changliu Liu. Robust online model adaptation by extended kalman filter with exponential moving average and dynamic multi-epoch strategy. *arXiv:1912.01790*, 2019.

Abulikemu Abuduweili, Siyan Li, and Changliu Liu. Adaptable human intention and trajectory prediction for human-robot collaboration. *arXiv preprint arXiv:1909.05089*, 2019.

Angelo Alessandri, Marta Cuneo, S Pagnan, and Marcello Sanguineti. A recursive algorithm for nonlinear least-squares problems. *Computational Optimization and Applications*, 38(2):195–216, 2007.

Brian DO Anderson and John B Moore. *Optimal filtering*. Courier Corporation, 2012.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

Shubhendu Bhasin, Rushikesh Kamalapurkar, Huyen T Dinh, and Warren E Dixon. Robust identification-based state derivative estimation for nonlinear systems. *IEEE Transactions on Automatic Control*, 58(1):187–192, 2012.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar F Zaidan. Findings of the 2010 joint workshop on statistical machine translation and metrics for machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 17–53. Association for Computational Linguistics, 2010.

Yujiao Cheng, Weiye Zhao, Changliu Liu, and Masayoshi Tomizuka. Human motion prediction using semi-adaptable neural networks. In *2019 American Control Conference (ACC)*, pages 4884–4890. IEEE, 2019.

James Colyar and John Halkias. Us highway 101 dataset. *Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030*, 2007.

John Cooper, Jiaxing Che, and Chengyu Cao. The use of learning in fast adaptation algorithms. *International Journal of Adaptive Control and Signal Processing*, 28(3-5):325–340, 2014.

Alexander Fink, Oliver Nelles, Martin Fischer, and Rolf Isermann. Non-linear adaptive control of a heat exchanger. *International Journal of Adaptive Control & Signal Processing*, 15(8):883–906, 2001.

Youji Iiguni, Hideaki Sakai, and Hidekatsu Tokumaru. A real-time learning algorithm for a multilayered neural network based on the extended kalman filter. *IEEE Transactions on Signal processing*, 40(4):959–966, 1992.

Andrew H Jazwinski. *Stochastic processes and filtering theory*. Courier Corporation, 2007.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jyrki Kivinen, Alexander J Smola, and Robert C Williamson. Online learning with kernels. *IEEE transactions on signal processing*, 52(8):2165–2176, 2004.

Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH journal*, 1(1):1, 2014.

Lennart Ljung and Pierre Priouret. A result on mean square error obtained using general tracking algorithms. *International Journal of Adaptive Control & Signal Processing*, 5(4):231–248, 2010.

Hiroyuki Moriyama, Nobuo Yamashita, and Masao Fukushima. The incremental gauss-newton algorithm with adaptive stepsize rule. *Computational Optimization and Applications*, 26(2):107–141, 2003.

Alex T Nelson. Nonlinear estimation and modeling of noisy time-series by dual kalman filtering methods. *Doctor of Philosopy, Oregon Graduate Institute of Science and Technology*, 2000.

Levent Ozbek and Murat Efe. An adaptive extended kalman filter with application to compartment models. *Communications in Statistics-Simulation and Computation*, 33(1):145–158, 2004.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12 (1):145–151, 1999.

Dennis W. Ruck, Steven K. Rogers, Matthew Kabrisky, Peter S. Maybeck, and Mark E. Oxley. Comparative analysis of backpropagation and the extended kalman filter for training multilayer perceptrons. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):686–691, 1992.

Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: A survey. *arXiv preprint arXiv:1905.06113*, 2019.

Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*, 2017.

Wenwen Si, Tianhao Wei, and Changliu Liu. Agen: Adaptable generative prediction networks for autonomous driving. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 281–286. IEEE, 2019.

Andreas S Weigend. *Time series prediction: forecasting the future and understanding the past*. Routledge, 2018.

Jann N Yang, Silian Lin, Hongwei Huang, and Li Zhou. An adaptive extended kalman filter for structural damage identification. *Structural Control and Health Monitoring: The Official Journal of the International Association for Structural Control and Monitoring and of the European Association for the Control of Structures*, 13(4):849–867, 2006.

Weiye Zhao, Liting Sun, Changliu Liu, and Masayoshi Tomizuka. Experimental evaluation of human motion prediction: Toward safe and efficient human robot collaboration, 2020.