On the suitability of incremental learning for regression tasks in exoskeleton control

Jonathan Jakob, Martina Hasenjäger, Barbara Hammer

2021

Preprint:

This is an accepted article published in IEEE Symposium on Computational Intelligence in Data Mining (CIDM). The final authenticated version is available online at: https://doi.org/[DOI not available]

On the suitability of incremental learning for regression tasks in exoskeleton control

1st Jonathan Jakob
Technical Faculty
Bielefeld University
Bielefeld, Germany
jjakob@techfak.uni-bielefeld.de

2nd Martina Hasenjäger
CGLP
Honda Research Institute Europe
Offenbach, Germany
martina.hasenjaeger@honda-ri.de

3rd Barbara Hammer
Technical Faculty
Bielefeld University
Bielefeld, Germany
bhammer@techfak.uni-bielefeld.de

***Abstract*—In recent times, a new generation of modern exoskeleton robots has come into existence, that aims to utilize machine learning to learn the specific needs and preferences of its users. A simple way to facilitate a personalization of an exoskeleton to the end user is to make use of incremental algorithms that keep learning throughout their deployment. However, it is not clear, if any standard algorithms are fast enough to keep pace with sudden change points in the data stream, like for example the change in movement pattern from a normal walk to going up the stairs. In this paper, we study how well common incremental regression algorithms are suited to predict such an ongoing data stream. We use both, theoretical benchmarks and real world human movement data, to evaluate how fast an algorithm reacts to change points in the data, and how well it is able to remember reoccurring patterns. The results show that a simple KNN algorithm outperforms all other more sophisticated models.**

***Index Terms*—incremental, online, time series, data streams, regression, exoskeleton**

I. Introduction

Lower body exoskeletons are assistive devices, designed to either help humans recover from injury or support them in work related tasks [1]. It has, for example, been shown, that lower body exoskeletons can reduce the metabolic cost of a human walking under load by up to 15% [2, 3]. The first of these exoskeletons were designed about twenty years ago and were purely mechanical devices that had to be fit to a user by hand. A newer generation of exoskeletons started to use electric motors as actuators, which are controlled through information gained from either *IMU* or *EMG* sensors. With them, so called human-in-the-loop fitting approaches came into play [4, 5]. These exoskeleton robots are able to adjust themselves to a human, but they rely on explicit user feedback, or the measuring of metabolic cost by external devices, to facilitate the adaption process. Nowadays, the newest generation of exoskeletons aims to utilize machine learning to automate the adaption to all kinds of users by directly learning their preferences and needs. Hereby, a wide range of different approaches are possible [6–12]. What they all have in common, is the need to accurately predict a user's next action, in order to provide adequate support for that specific movement. Furthermore, they should be able to adapt to any kind of potential end user with minimal supervision through another human.

Incremental or online learning is a machine learning paradigm in which a model is updated after each instance of data that is fed into it. This paradigm is especially suited for large data sets, that are too big to be processed in batch fashion, or in situations where data becomes available only instance after instance, as e.g. in potentially infinite streams of data. Incremental learning has made great strides in recent years [13] but none of these approaches have yet been applied to human movement prediction for exoskeletons. Given the need of exoskeletons to adapt to specific end users and the fact that movement prediction is dependent on continuously advancing data streams, a quick incremental regression algorithm could be ideally suited to predict the data stream coming from given *IMU* or *EMG* sensors in order to facilitate exoskeleton control. Since, the algorithm is incremental, it can adapt itself to any user. However, it is not clear, whether a standard incremental algorithm would be quick enough to adequately react to a sudden change in the movement pattern, such as e.g. the transition from a normal walk to going down the stairs.

In this contribution we are one of the first to systematically investigate a wide range of standard incremental online regression algorithms with regard to their behaviour around change-points in streaming data. Furthermore, we examine, how these algorithms react when previously seen patterns occur again in the data stream, because algorithms with good recalling abilities should naturally handle a reoccurring change-point event even quicker than before. In that, our problem is related to concept drift, especially to the notions of abrupt and reoccurring drift as defined in [14].

This paper is structured in the following way. The next section defines our problem mathematically. In section III, we give a brief overview of the algorithms that we compare. After that, we introduce a few new benchmark data sets together with an explanation of our experimental setup. Then, in section V, we present the results, followed by a conclusion in section VI.

II. Overarching Challenge

We use incremental regression to predict a data stream one instance after another. Hereby, we define a data stream $S = \{s_1, s_2, s_3, ..., s_t\}$ as a potentially infinite set of data points $s_i \in \mathbb{R}^n$.

An incremental model is an algorithm that receives a data stream instance after instance and generates a sequence of models $h_1, h_2, h_3, ..., h_t$ where $h_{i-1}(s_i) = s_{i+1}$ is a function that acts on the current instance and predicts the value of the next instance of the data stream. After that, the true value s_{i+1} is revealed and a new model h_i is learned.

To evaluate this regression task, the *Interleaved train test error (ITTE)* is applied:

$$E(S) = \sqrt{\frac{1}{t}\sum_{i=1}^{t}(h_{i-1}(s_i) - s_{i+1})^2}$$

This ITTE measures the *Root Mean Squared Error (RMSE)* over every model h_i up to a given time point t.

III. Models

The incremental regression algorithms that we compare can be divided into two classes: recurrent and non-recurrent models. In the recurrent category, we concentrate on four standard algorithms with different memorizing capabilities: The ESN [15], the GRU [16], the LSTM [17] and the NRU [18]. In the other class, we test a KNN [19], a random forest as a representative for tree based methods [20], Vector autoregression (VAR) as a representative for linear models [21], a SVR [22] and a simple naive predictor as a baseline. For the ESN and the baseline we used our own implementation. The LSTM, GRU, NRU and VAR models were implemented using PyTorch [23]. For the SVR we used the Scikit-Learn [24] implementation, and the other models where taken from the online library River [25]. In addition to that, we also performed tests on the Hoeffding-Tree and Linear model from [25] as well as on a Transformer [26], a LMU [27] and on Gaussian process regression [28]. However, all of these algorithms performed consistently worse than the ones we concentrate on below, with the Transformer not being suited for this type of problem at all.

A. Recurrent Models

Echo State Network (ESN) [15]: The ESN is an architecture of the reservoir computing family which utilises a random dynamical reservoir to map an input signal into high dimensional space. From there, output units are trained with linear regression, rendering the whole approach very cost efficient. ESNs have the property that all input information is guaranteed to be washed out of the reservoir after a certain time. This enables them to be agile learners in the face of suddenly changing input data but also constrains their abilities with regard to the long term recalling of information.

Gated Recurrent Unit (GRU) [16]: A GRU is a recurrent architecture that uses two internal gates to actively manage the information in its internal cell state. GRUs are trained with back-propagation through time, which renders their computational cost much higher than that of ESNs. However, due to their higher internal complexity, they are also much better adept to store and recall information over longer periods of time.

Long Short Term Memory (LSTM) [17]: LSTMs are deep recurrent architectures. Compared to the GRU, they use one more internal gate and two distinct cell states to manage the incoming information. This renders them even more computationally expensive, although in theory they are also a little more powerful when it comes to long term dependencies.

Non-saturating Recurrent Unit (NRU) [18]: The final recurrent architecture that we compare is the NRU. This unit relies on a memory mechanism, which forgoes saturating activation functions as well as saturating internal gates, in order to alleviate the problem of vanishing gradients. Thus, the NRU is specifically designed to model long-term dependencies.

B. Non-recurrent Models

K-nearest neighbors (KNN) [19]: Knn regression is a simple algorithm that predicts a new sample by the mean value of the outputs at the *k* closest neighbors in its memory.

Random Forest (RF) [20]: Random forests are an ensemble learning technique that utilize a multitude of internal decision trees for the regression task. Each decision tree computes its own solution to the problem at hand and the final value is given by the mean solution of all trees in the forest.

Vector Autoregression (VAR) [21]: The VAR model is a statistical model that generalizes univariate autoregressive models by allowing for multivariate time series. It predicts the upcoming sample of a series as a linear combination of several lag variables along the series at hand.

Support Vector Regression (SVR) [22]: The SVR is a regressive version of the well known SVM. It utilizes support vectors to fit the error inside a certain threshold, meaning that it approximates the best value within a given margin.

Naive Predictor (NP): The final model that we use is a simple naive predictor, that forecasts an upcoming value by assigning the value of the current time point. This model is not meant to really compete with the others but simply acts as a baseline for the regression task.

IV. Experiments

We conduct three distinct experiments to test how well any of the listed models is suited for our problem. To perform the first two experiments we created four new benchmark data sets from chaotic systems. The last experiment is conducted on real world data. The data sets are characterized in the next subsection. Then, we list the hyperparameters of the models, and afterwards, the setup of the experiments themselves is described in detail.

A. Chaotic Data

To test the algorithms with regard to our problem, we need data that is sufficiently difficult to predict in order to pose enough of a challenge for clear differences in the models

to show themselves. Therefore, we chose to create special benchmark data sets from chaotic systems. We use the Lorenz system [29] along with the Roessler system [30] as well as the Tinkerbell map [31] along with the Duffing map [32] to create our four new benchmark data sets. Hereby, the Lorenz and Roessler systems are three dimensional, while the Tinkerbell and Duffing map are two dimensional data streams. Figure 1 shows the expansion of the first 100 data points in a stream created from each of the chaotic systems. Our benchmark data sets are created from the raw streams in the following way. Data streams holding 2000 instances of the three dimensional Lorenz and Roessler systems are glued together in alternate fashion, to form a data set with two sudden change points of which the latter initiates the reoccurring of the first system for a second time. The same is done for the two dimensional Tinkerbell and Duffing maps, leading to four data sets of 6000 data points each. Figure 2 shows a sample plot of the Roessler-Lorenz data set.

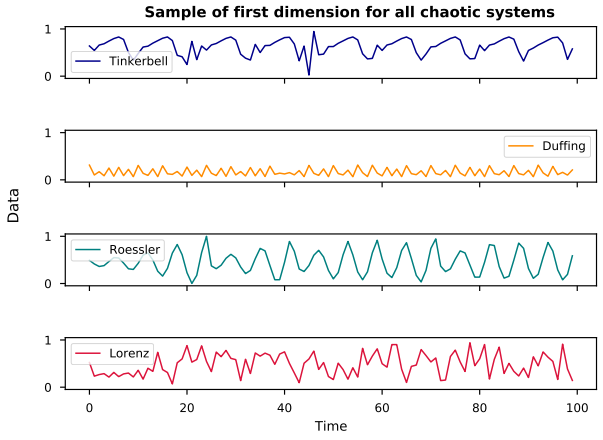

Fig. 1. Example plot of the first 100 data points in the first dimension for all chaotic systems. The first row shows the Tinkerbell map, the second row shows the Duffing map, the third row shows the Roessler system and the last row shows the Lorenz system.

B. Real World Data

We use data from the Human Gait Database (HuGaDB) [33] to evaluate the algorithms on the real world scenario of a person alternating between the movements walking and going up the stairs. HuGaDB comprises a wide range of data sets recorded from multiple people performing all kinds of standard human movements. We use a subset of four randomly chosen people going up a stairway. These data sets have 13 alternating segments of the movements walking and going up the stairs. The data was recorded by six *IMU* and two *EMG* sensors leading to 38 dimensions in total.

C. Hyperparameters

We used the following robust hyperparameters for all data sets. An ESN with 100 neurons, input scaling of 0.4, spectral radius of 1.1 and a leaking rate of 0.4. A GRU and an LSTM

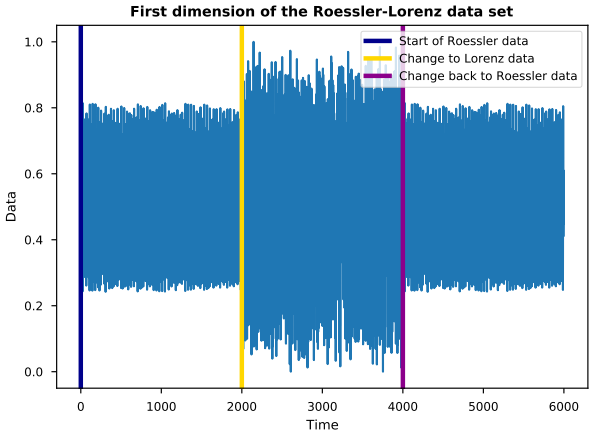

Fig. 2. Example plot of the first dimension of the Roessler-Lorenz data set. The first 2000 samples are taken from the Roessler system, followed by another 2000 points from the Lorenz system. In the end another change point back to the Roessler system is induced.

with 100 neurons each and a dropout layer with a probability of 0.4. A NRU with 100 neurons and standard parameters as suggested by the authors. A KNN with five neighbors and a memory size of 6000 for the first two experiments and 500 for the last experiment. A RF model using Hoeffding-Trees and standard parameters as given by River [25], a SVR with standard parameters as given by Scikit-Learn and finally, a VAR model with five lag variables. All computations were performed on the same Intel Xeon W-2145 CPU at a clock rate of 3.7 GHz.

D. Chaotic-Data Experiment 1

In the first experiment, we focus on the first change point in the data sets. The objective is to find out, how fast a given algorithm reacts to a sudden change of pattern inside a data stream. We quantify this by measuring the area under the curve of accumulated mean errors. That is, we measure how fast the mean error decreases to its final value at the end of the 2000 point segment by the size of the area under the curve of mean errors taken along every point of the segment. Mathematically, the area under the curve (AuC) is defined as follows:

$$\frac{1}{t}\sum_{i=1}^{t}\sqrt{\frac{1}{i}\sum_{j=1}^{i}(h_{j-1}(s_j)-s_{j+1})^2}$$

where t is the length of the segment over which the AuC is computed. Figure 3 shows a depiction of the AuC for the Lorenz-Roessler data set. We conduct this experiment for three distinct time horizons: 1-step, 5-step and 10-step prediction. With 1-step predictions being the standard case, we opt to also use 5-step ahead as well as 10-step ahead predictions in order to simulate the prediction of one or two physical walking steps with an exoskeleton. These multi-step ahead predictions are facilitated by recursively feeding a 1-step prediction back

into the algorithms, thereby using them as pattern generators instead of pure predictors.

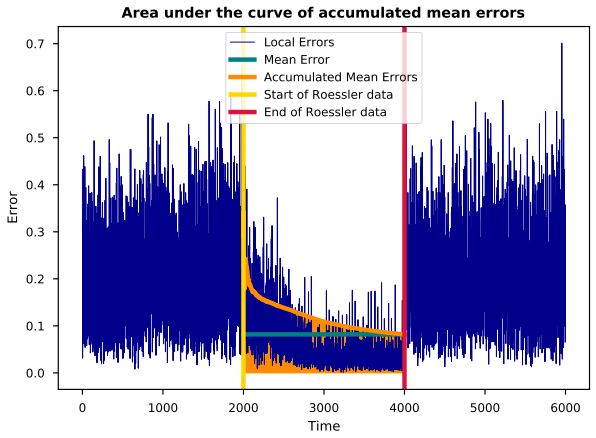

Fig. 3. Example plot of the area under the curve of accumulated mean errors. Each point on the orange curve is the mean error from the start of the segment up to that point. The green line depicts the mean error over the whole segment, meaning that the green line and the orange curve converge at the end of the segment by definition. The curve of accumulated mean errors decreases with time. The area under that curve gives a measure of how fast they decrease, relative to their overall level.

E. Chaotic-Data Experiment 2

In the second experiment, the objective is to find out which algorithm has the best memorizing capabilities. The reasoning behind this is simple. An algorithm, that is capable of remembering a previously seen pattern over a long period of time is more likely to react faster to its recurrence than without remembering it. We therefore quantify this capability as the reduction in the 1-step *Root Mean Square Error (RMSE)* over the whole segment from the first occurrence to the second one. Note, that we do not compare the 5-step and 10-step RMSEs as well because the capacity to remember concepts is independent from the prediction horizon.

F. Real-World-Data Experiment

In the last experiment, we want to see how the algorithms behave in a high-dimensional real world application. Since, the HuGaDB data sets hold 13 alternating segments of movement data we do not evaluate all change point events separately. Instead, we compare the algorithms via the total RMSE over the whole data sets, again for a 1-step, 5-step and 10-step time horizon.

V. Results

In this section, we report the results of our experiments, and in the end, we give an account of the time wise behaviour of each algorithm.

A. Chaotic-Data Experiment 1

The results of the first experiment are shown in Table I. The KNN algorithm wins the 1-step prediction in three out of four data sets, only to be overtaken once by the RF algorithm by a very small margin. In the 5-step prediction, the KNN wins two of the four data sets. The other two wins go to the RF and NRU models, again by very small margins. In the 10-step prediction, the KNN wins only one data set, with the RF algorithm winning two and the NRU winning one. Overall, the KNN performs best but with a rising prediction horizon, the RF and NRU manage to slowly overtake it. For the recurrent models, one can observe, that the ESN 1-step performance is on par with the GRU and the LSTM but this performance deteriorates rapidly for higher time horizons. In fact, the ESN is the only recurrent model, whose 10-step prediction is so bad that it falls under the baseline. The GRU, LSTM and the NRU exhibit a somewhat stable behaviour with regard to rising time horizons but overall the NRU consistently beats the GRU and LSTM by a definite margin. No virtual difference can be observed between the GRU and the LSTM. For the non-recurrent models, the KNN is the clear winner but the RF comes very close in performance. The VAR model and the NP baseline are consistently worse than the KNN and RF models. The worst performer is the SVR, falling under the baseline often even for a 1 step prediction horizon.

B. Chaotic-Data Experiment 2

The results of the second experiment are shown in Figure 4. The plots show the RMSE for the first and the second occurrence of a segment in the data sets. It can be seen, that the KNN consistently beats all other algorithms in terms of RMSE, closely followed by the RF model. The baseline is the worst performer on the Lorenz-Roessler data set but is exceeded by the VAR once and SVR twice on the other sets. The recurrent models usually perform in the same range but sometimes the ESN takes a slight lead. The GRU and the LSTM again perform the same way but are slightly beaten by the NRU.

Coming to the difference in RMSE between the first and second occurrence of the respective data segments, one can observe that the KNN is the clear winner here as well. One has to remember though, that our model has a memory that is big enough for the whole data sequence. Applying this in practice, where data streams are potentially infinite is a subject of our ongoing research. The second best algorithm with regard to remembering is the RF model. It performs worse than the KNN on all sets but also better than all other algorithms. The VAR model usually has a higher remembrance than the recurrent models but its general RMSE is also higher, taking it out of contest. Of the recurrent models, the NRU showcases the best remembering capabilities but the overall performance is not enough to come even close to the KNN or the RF.

C. Real-World-Data Experiment

The results of the final experiment are shown in Table II. Here, the KNN unilaterally wins on all data sets and for all time horizons, although it is closely followed by the RF model. Peculiarly, it can be observed, that the 5-step horizon usually leads to better results than the 1-step prediction. Interestingly, the ESN now becomes the third best performing model in

TABLE I

AREA UNDER THE CURVE FOR THE SEGMENT AFTER THE FIRST CHANGE POINT EVENT IN ALL DATA SETS, FOR ALL CLASSIFIERS AND FOR 1-STEP, 5-STEP AND 10-STEP TIME HORIZONS. ALL VALUES ARE AVERAGED OVER 10 INDEPENDENT RUNS. THE VARIANCES ARE LOW EXCEPT FOR THE 10-STEP FORECASTING OF THE ESN CLASSIFIER. THE BEST PERFORMANCE PER DATA SET AND FORECASTING HORIZON IS MARKED IN BOLD.

	Data sets and number of forecasting steps											
	Tinkerbell-Duffing			**Duffing-Tinkerbell**			**Lorenz-Roessler**			**Roessler-Lorenz**		
Algorithms	**1**	**5**	**10**	**1**	**5**	**10**	**1**	**5**	**10**	**1**	**5**	**10**
ESN	0.108	0.154	2.811	0.124	0.145	7.659	0.121	0.209	9.953	0.234	0.225	3.246
GRU	0.117	0.128	0.138	0.119	0.120	0.137	0.108	0.101	0.111	0.219	0.213	0.217
LSTM	0.126	0.125	0.135	0.115	0.116	0.128	0.113	0.104	0.112	0.219	0.210	0.217
NRU	0.084	0.104	0.124	0.104	0.105	0.124	0.076	0.077	0.104	0.211	**0.208**	**0.216**
KNN	**0.049**	**0.054**	**0.098**	0.059	0.093	0.127	**0.066**	**0.076**	0.108	**0.156**	0.229	0.259
RF	0.060	0.101	0.146	**0.057**	**0.086**	**0.120**	0.073	0.081	**0.099**	0.185	0.221	0.233
VAR	0.106	0.114	0.134	0.137	0.135	0.133	0.114	0.102	0.115	0.230	0.219	0.219
SVR	0.201	0.240	0.251	0.140	0.174	0.200	0.159	0.156	0.159	0.248	0.239	0.242
NP	0.167	0.140	0.145	0.145	0.160	0.172	0.111	0.133	0.136	0.279	0.264	0.268

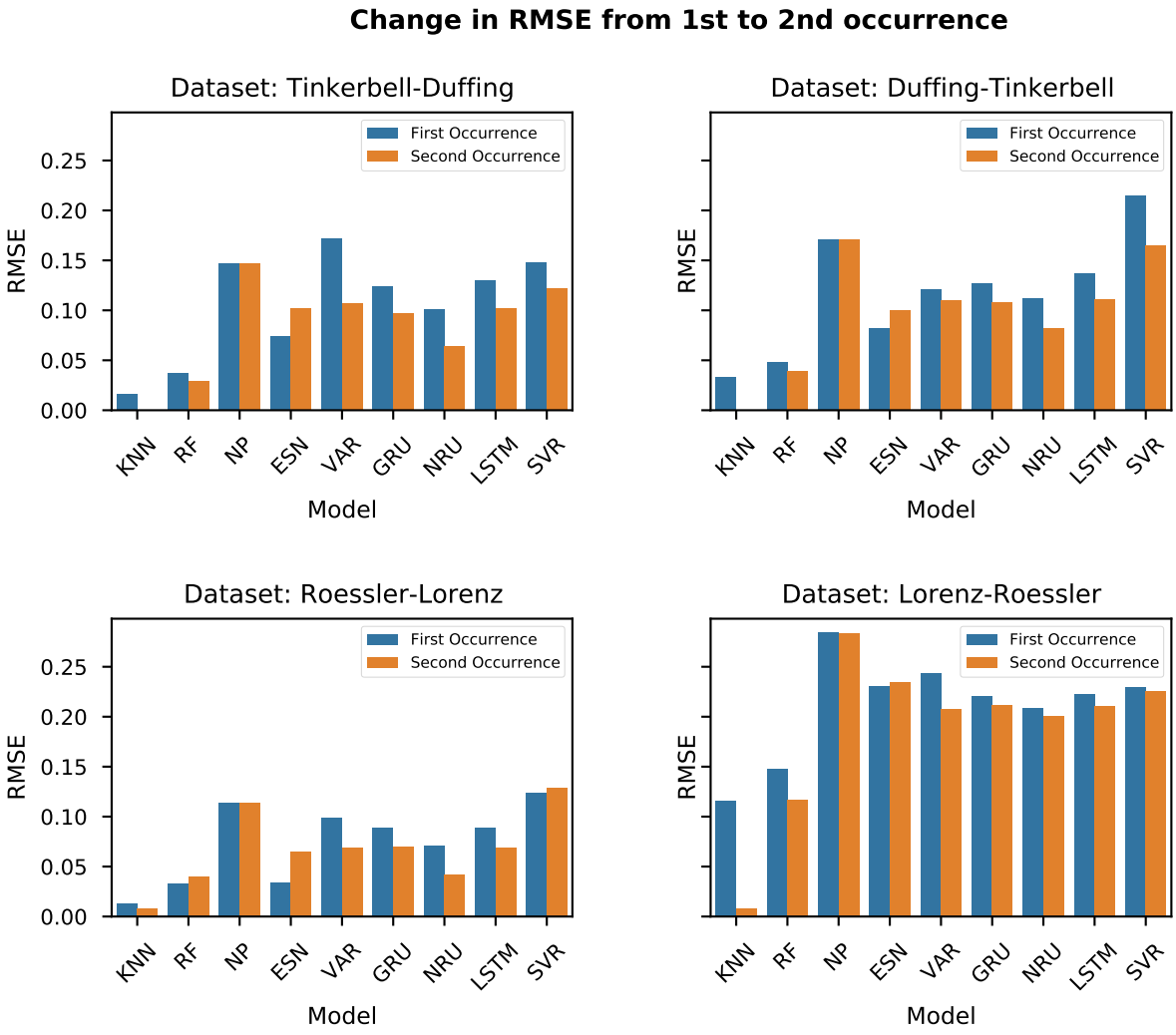

Fig. 4. Change in RMSE for a 1-step time horizon from the first to the second occurrence for all data sets and algorithms.

terms of 1-step predictions but it loses ground for higher time horizons, while the KNN and RF remain more stable in those scenarios. The NRU performs slightly better than the GRU and the LSTM in the first and second time scenario but it is overtaken by the others in the 10-step time horizon. However, it has to be noted that all recurrent models with the exception of the ESN now fall under the baseline by a definite margin. The VAR is the worst performing model overall while the SVR is better than the recurrent networks but still worse than the baseline. In total, the baseline clearly separates the KNN, RF and ESN models from the rest in this experiment.

D. Time

In this section we briefly look at the computation times of all algorithms to evaluate which of them are fast enough to handle an incremental task in the real world. The computation times for a full 10-step ahead prediction sweep through all the data sets are shown in Table III. This means, that the total computation time for 60000 predictions on a data set with 6000 instances is evaluated. The fastest algorithms are unfortunately also those that did not perform very well in the experiments. However, the KNN, and the NRU are still sufficiently fast on the 6000 point sets to be applied in a real world scenario. The fastest models are the ESN and the VAR. On the other hand, the RF algorithm in the implementation that we used is far too slow to be applied online. Still, it has to be noted, that this particular implementation does not have multi core support, so faster implementations might be possible.

VI. Conclusion

In this paper, we examined how well a range of popular online regression algorithms are applicable to the problem of modeling change point events in movement data for exoskeletons. To test this, we conducted three experiments with a clear outcome. Out of all tested models, the KNN is the by far best performing one. The RF model can almost keep up, but its exorbitant computation times render it useless for practical deployment. Of the recurrent models, the NRU performed best on the chaotic data, but it cannot match the performance of the KNN at all. When the data gets easier, as in the real world example, the ESN takes the lead in the recurrent category, but it too, cannot compete with the KNN. The one problem remaining is, that we used a KNN with enough memory to hold the whole data sequences in the chaotic experiments. Since, this is inapplicable in the real world, where data streams are potentially infinite, building a system with a limited KNN memory but similar performance is desirable. One such system is the SAMkNN [34] framework. However, when we tested it, the performance was worse than that of a standard KNN, especially in the real world experiment. This is due to the fact, that only virtual but no real drift occurs in these data sets, meaning, that data from one movement type does not interfere with data from other movements. In this case, the memory management of SAMkNN is not a good fit, as it works best for real drift scenarios where the internal memories are reduced to only hold the current concepts of the data. Thus, finding another memory management system that does not exclude old patterns remains the subject of future work.

References

[1] Pamungkas, D., Caesarendra, W., Soebakti, H., Analia, R., Susanto, S. (2019). Overview: Types of Lower Limb Exoskeletons. Electronics, 8, 1283.

[2] Shepertycky, M., Burton, S., Dickson, A., Liu, Y.F., Li, Q. (2021). Removing energy with an exoskeleton reduces the metabolic cost of walking. Science, 372(6545), 957–960.

[3] Mooney, L.M., Rouse, E.J. Herr, H.M. Autonomous exoskeleton reduces metabolic cost of human walking during load carriage. J NeuroEngineering Rehabil 11, 80 (2014).

[4] Zhang, J., Fiers, P., Witte, K., Jackson, R., Poggensee, K., Atkeson, C., Collins, S. (2017). Human-in-the-loop optimization of exoskeleton assistance during walking. Science, 356(6344), 1280–1284.

[5] Li, Z., Zhao, K., Zhang, L., Wu, X., Zhang, T., Li, Q., Li, X., Su, C.Y. (2020). Human-In-the-Loop Control of a Wearable Lower Limb Exoskeleton for Stable Dynamic Walking. IEEE/ASME Transactions on Mechatronics, 1-1.

[6] Thakur, Nirmalya and Han, Chia Y., Exoskeleton-Based Multimodal Action and Movement Recognition: Identifying and Developing the Optimal Boosted Learning Approach (June 12, 2021). Journal of Advances in Artificial Intelligence and Machine Learning. 2021; Volume 1, Issue 1, Article 4.

[7] Zhao, C.y., Zhang, X.g., Guo, Q. (2012). The Application of Machine-Learning on Lower Limb Motion Analysis in Human Exoskeleton System. Social Robotics, 600–611.

[8] N. Amamcherla, A. Turlapaty and B. Gokaraju, "A Machine Learning System for Classification of EMG Signals to Assist Exoskeleton Performance," 2018 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), 2018, pp. 1-4.

[9] Masashi Hamaya, Takamitsu Matsubara, Tomoyuki Noda, Tatsuya Teramae, Jun Morimoto (2017). Learning assistive strategies for exoskeleton robots from user-robot physical interaction. Pattern Recognition Letters, 99, 67-76.

[10] Wang, Can and Wu, Xinyu and Ma, Yue and Wu, Guizhong and Luo, Yuhao. (2018). A Flexible Lower Extremity Exoskeleton Robot with Deep Locomotion Mode Identification. Complexity. 2018. 1-9.

[11] Jin, Xin and Guo, Jia and Li, Zhong and Wang, Ruihao. (2020). Motion Prediction of Human Wearing Powered Exoskeleton. Mathematical Problems in Engineering. 2020. 1-8.

[12] J. -L. Ren, Y. -H. Chien, E. -Y. Chia, L. -C. Fu and J. -S. Lai, "Deep Learning based Motion Prediction for Exoskeleton Robot Control in Upper Limb Rehabilitation," 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 5076-5082.

TABLE II

ROOT MEAN SQUARED ERRORS OVER THE WHOLE DATA SETS FOR ALL CLASSIFIERS AND FOR 1-STEP, 5-STEP AND 10-STEP TIME HORIZONS. ALL VALUES ARE AVERAGED OVER 10 INDEPENDENT RUNS. THE BEST PERFORMANCE PER DATA SET AND FORCASTING HORIZON IS MARKED IN BOLD.

	Data sets and number of forecasting steps											
	Person 1			**Person 2**			**Person 3**			**Person 4**		
Algorithms	**1**	**5**	**10**	**1**	**5**	**10**	**1**	**5**	**10**	**1**	**5**	**10**
ESN	0.067	0.072	0.126	0.081	0.084	0.193	0.068	0.076	0.200	0.067	0.074	0.132
GRU	0.121	0.118	0.140	0.119	0.115	0.135	0.118	0.117	0.142	0.119	0.114	0.134
LSTM	0.117	0.112	0.129	0.121	0.115	0.127	0.120	0.117	0.138	0.123	0.119	0.136
NRU	0.108	0.109	0.145	0.112	0.125	0.173	0.111	0.123	0.187	0.109	0.114	0.161
KNN	**0.058**	**0.055**	**0.068**	**0.066**	**0.061**	**0.075**	**0.060**	**0.056**	**0.070**	**0.055**	**0.050**	**0.065**
RF	0.060	0.059	0.081	0.071	0.071	0.095	0.070	0.067	0.093	0.059	0.058	0.079
VAR	0.244	0.448	6.358	0.198	0.366	4.975	0.318	1.052	9.716	0.205	0.705	6.665
SVR	0.104	0.095	0.104	0.105	0.094	0.102	0.105	0.101	0.112	0.101	0.091	0.100
NP	0.074	0.083	0.107	0.085	0.092	0.115	0.074	0.085	0.114	0.067	0.072	0.099

TABLE III

COMPUTATION TIME OF A FULL RUN WITH 10-STEP FORECASTING HORIZON FOR ALL DATA SETS AND ALGORITHMS IN SECONDS. EACH SET HOLDS 6000 DATA POINTS, MEANING 60000 PREDICTIONS WERE MADE THROUGH THE 10-STEP FORECASTING HORIZON.

	Algorithms								
Data set	**ESN**	**GRU**	**LSTM**	**NRU**	**KNN**	**RF**	**VAR**	**SVR**	**NP**
Tinkerbell-Duffing	6	36	42	357	250	2.8×10^4	11	587	1
Duffing-Tinkerbell	7	35	42	340	249	2.3×10^4	11	755	1
Lorenz-Roessler	8	35	47	337	381	4.2×10^4	11	1.0×10^3	1
Roessler-Lorenz	6	36	47	349	380	3.8×10^4	11	1.1×10^3	1

[13] Gomes, Heitor Murilo, Read, Jesse, Bifet, Albert, Barddal, Jean Paul, Gama, Joao. (2019). Machine learning for streaming data: state of the art, challenges, and opportunities. ACM SIGKDD Explorations Newsletter. 21. 6-22.

[14] Zliobait, I., Pechenizkiy, M., Gama, J. (2016). An Overview of Concept Drift Applications in Big Data Analysis: New Algorithms for a New Society. pp. 91-114.

[15] Jaeger, H., Haas, H. (2004). Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. Science (New York, N.Y.), 304, 78-80.

[16] Cho, K., Merriënboer, B., Gulcehre, C., Bougares, F., Schwenk, H., Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. ArXiv Preprint ArXiv:1406.1078.

[17] Hochreiter, S., Schmidhuber, J. (1997). Long Short-Term Memory. Neural Comput., 9(8), 1735–1780.

[18] Chandar, S., Sankar, C., Vorontsov, E., Kahou, S., Bengio, Y. (2019). Towards Non-Saturating Recurrent Units for Modelling Long-Term Dependencies. Proceedings of the AAAI Conference on Artificial Intelligence, 33, 3280-3287.

[19] E. Fix, J. L. Hodges (1989). Discriminatory Analysis - Nonparametric Discrimination: Consistency Properties. International Statistical Review, 57, 238.

[20] Tin Kam Ho (1995). Random decision forests. In Proceedings of 3rd International Conference on Document Analysis and Recognition (pp. 278-282 vol.1).

[21] Luetkepohl, H. (2011). Vector Autoregressive Models. International Encyclopedia of Statistical Science, 1645–1647.

[22] Drucker, H., Burges, C., Kaufman, L., Smola, A., Vapnik, V. (1996). Support Vector Regression Machines. In Proceedings of the 9th International Conference on Neural Information Processing Systems (pp. 155–161). MIT Press.

[23] Adam Paszke and Sam Gross and Francisco Massa and Adam Lerer and James Bradbury and Gregory Chanan (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. CoRR.

[24] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. (2011). Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), 2825–2830.

[25] Jacob Montiel and Max Halford and Saulo Martiello Mastelini and Geoffrey Bolmier and Robin Vaysse and Albert Bifet (2020). River: machine learning for streaming data in Python. CoRR.

[26] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. (2017). Attention is All You Need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (pp. 6000–6010). Curran Associates Inc..

[27] Aaron R. Voelker, Ivana Kajic, Chris Eliasmith (2019). Legendre Memory Units: Continuous-Time Representation in Recurrent Neural Networks. In Advances in Neural Information Processing Systems (pp. 15544–15553).

[28] Williams, C., Rasmussen, C. (1995). Gaussian Processes for Regression. In Proceedings of the 8th International

Conference on Neural Information Processing Systems (pp. 514–520). MIT Press.
[29] Lorenz, Edward Norton (1963). Deterministic non-periodic flow. Journal of the Atmospheric Sciences. 20 (2): 130–141.
[30] Rössler, O. E. (1976), An Equation for Continuous Chaos, Physics Letters, 57A (5): 397–398.
[31] Nusse, H. E. and Yorke, J. A. (1997) Dynamics: Numerical Explorations (Springer, NY)
[32] G. Duffing, Erzwungene Schwingungen bei Veränderlicher Eigenfrequenz., F. Vieweg u. Sohn, Braunschweig, 1918.
[33] Chereshnev, R., Kertesz-Farkas, A. (2017). HuGaDB: Human Gait Database for Activity Recognition from Wearable Inertial Sensor Networks. In International Conference on Analysis of Images, Social Networks and Texts (pp. 131–141).
[34] Losing, V., Hammer, B., Wersing, H. (2016) KNN Classifier with Self Adjusting Memory for Heterogeneous Concept Drift. IEEE 16th International Conference on Data Mining (pp. 291-300).