Online Prediction of Exponential Decay Time Series with Human-Agent Application

Ariel Rosenfeld[1], **Joseph Keshet**[2], **Claudia V. Goldman**[3], **Sarit Kraus**[4]

Abstract. Exponential decay time series are prominent in many fields. In some applications, the time series behavior can change over time due to a change in the user's preferences or a change of environment. In this paper we present an innovative online learning algorithm, which we name *Exponentron*, for the prediction of exponential decay time series. We state a regret bound for our setting, which theoretically compares the performance of our online algorithm relative to the performance of the best batch prediction mechanism, which can be chosen in hindsight from a class of hypotheses after observing the entire time series. In experiments with synthetic and real-world data sets, we found that the proposed algorithm compares favorably with the classic time series prediction methods by providing up to 41% improvement in prediction accuracy. Furthermore, we used the proposed algorithm for the design of a novel automated agent for the improvement of the communication process between a driver and its automotive climate control system. Throughout extensive human study with 24 drivers we show that our agent improves the communication process and increases drivers' satisfaction, exemplifying the Exponentron's applicative benefit.

1 Introduction

Exponential decay functions are popular in modeling real-world phenomena. For example, the decrease in radioactivity levels of radioactive substances, the cooling of an object in a cold environment [24] and human decline of memory retention over time [13] are all assumed to decay exponentially over time, and are usually modeled using exponential decay functions.

There are several learning algorithms for the prediction of exponential decay time series such as the Bayesian [6], regressive [12] or autoregressive [9] methods based on labeled training data (i.e, set of observed time series). However, most of these algorithms introduce two limitations: First, the algorithms are batch in nature and need to get the whole training set in advance. As such, *batch* algorithms cannot adapt to changes over time in the modeled phenomenon which may occur after the initial training phase, for example when modeling human preferences over time. Second, the algorithms are *general purpose*, and do not exploit the exponential decay nature of the time series. While a few online learning versions of the above models have been investigated recently [20], to the best of our knowledge none have specifically addressed exponential decay time series.

In this paper we present an online learning algorithm, which we call *Exponentron*, for the prediction of exponential decay time series. The online learning algorithm takes place in a sequence of consecutive rounds. In each round, the learner first receives a time series instance. Then, the learner is required to predict its parameters. At the end of the round, the learner obtains the correct parameters, and uses this information to improve its future predictions.

The Exponentron algorithm focuses on a special hypothesis family capturing the assumed exponential decay behavior of time series. It is aimed at optimizing the square loss function, and like other online learning algorithms it does not require any training data before deployment.

We state a regret bound for the Exponentron algorithm. Regret bounds are common in the analysis of online learning algorithms. A regret bound measures the performance of an online algorithm relative to the performance of the best competing hypothesis, which can be chosen in hindsight from a class of hypotheses, after observing the entire time series.

Empirically, we show that the algorithm significantly outperforms classic time series prediction methods' accuracy in predicting assumed exponential decay time series in two repeated real-world settings by up to 41%. Exponentron is then used for the enhancement of driver-automotive climate control system (CCS) interaction. A novel intelligent agent for Natural Interaction with humans in CCS using the Exponentron algorithm, which we named the NICE agent, is presented. The NICE agent was extensively evaluated with 24 human drivers in hot summer conditions. The agent successfully reduced the number of interactions needed by the driver to achieve her desired comfort state by 19% compared to state-of-the-art CCSs. The agent is also shown to achieve high driver satisfaction.

This paper makes the following contributions; (1) We propose a novel online learning algorithm, named *Exponentron*, for the prediction of exponential decay time series. The algorithm is the first of its kind as it is tailored to a unique class of time series. We provide both theoretical analysis and empirical evaluation of the algorithm. (2) We present a novel intelligent agent, named the *NICE* agent, that provides the driver with intelligent, natural interface with which she can change the parameters of her automotive climate control system (CCS). The agent reduces the driver's need for interaction with the CCS, and increases the driver's subjective satisfaction.

2 Time Series Preliminaries

A time series $\boldsymbol{s} = (s_0, s_1, \ldots, s_T)$ is an ordered sequence of values measured in equally spaced time intervals, where $s_t \in \mathbb{R}$ denotes an element at time t, $0 \leq t \leq T$. In this work, we assume that the time series was created by an exponential decay process. Assuming $(s_0, s_1, \ldots, s_{t-1})$ is the beginning of a series, the prediction of the next element is denoted $\hat{s}_t$.

[1] Bar Ilan University, Israel, email: arielros1@gmail.com
[2] Bar Ilan University, Israel.
[3] General Motors Advanced Technical Center, Herzliya, Israel.
[4] Bar Ilan University, Israel.

Intelligent agents often use time series prediction to improve their decision-making. A few recent examples include the repositioning of bikes in bike-sharing systems based on the prediction of bike usage [25], maintenance scheduling based on the prediction of ongoing game scores [31] and the prediction of passenger demand for better taxi routing [22]. A common theme among these systems is the use of classical, *general purpose* time series prediction methods as the basis for their *domain-specific* proposed approach and design.

Among the most commonly applied techniques for forecasting the continuation of a time series given its beginning are the autoregressive (AR) model, the autoregressive-moving average (ARMA) model (see [9] for a review) and the exponential smoothing (ES) forecasting method (see [15] for a review).

The autoregressive (AR) model generates its prediction using the following equation:

$$\hat{s}_t = c + \sum_{i=1}^{p} \varphi_i s_{t-i} \tag{1}$$

where c, p and φ_i are parameters of the model.

An AR model is in fact a linear regression of the current value of the time series against one or more prior values of the time series. The value of p is called the order of the AR model. The most commonly applied AR method, which is also used in this study, uses $p = 1$. This model is sometimes denoted $AR(1)$.

A popular extension of the AR model uses a moving average (MA), resulting in the autoregressive-moving average (ARMA) model [9]. ARMA is also known as the Box-Jenkins Approach. The ARMA model generates its prediction using the following equation:

$$\hat{s}_t = c + \sum_{i=1}^{p} \varphi_i s_{t-i} + \sum_{i=1}^{q} \theta_i e_{t-i}. \tag{2}$$

where $e_j = s_j - \hat{s}_j$ and $c, p, q, \varphi_i, \theta_i$ are parameters of the model.

An $ARMA$ model is in fact a linear regression of the current value of the time series against one or more prior values of the time series and one or more prior noise terms. The value of p is called the order of the AR part of the model and q is the order of the MA part of the model. The most commonly applied ARMA method, which is also used in this study, uses $p = q = 1$. This model is sometimes denoted $ARMA(1, 1)$.

Another prediction method is the exponential smoothing (ES) scheme (also known as the exponentially weighted moving average), which weighs past elements of the time series using exponentially decreasing weights. The most suitable exponential smoothing method, which is used in this study, is the *double* exponential smoothing method, denoted ES. ES forecasts the continuation of a time series using the following equations:

$$\begin{aligned} \hat{s}_t &= \alpha s_{t-1} + (1-\alpha)(\hat{s}_{t-1} + b_t) \\ b_t &= \gamma(\hat{s}_t - \hat{s}_{t-1}) + (1-\gamma) b_{t-1} \end{aligned} \tag{3}$$

where $0 < \alpha \leq 1$ and $0 < \gamma \leq 1$.

There are several methods for choosing $\hat{s_0}$ and b_0. The most common one, which is also used in this study, is $\hat{s_0} = s_0$ and $b_0 = 0$. Note that *single* exponential smoothing does not fit time series which present trends, and therefore is unsuitable for this study. *Triple* exponential smoothing, also known as the Holt-Winters exponential smoothing technique [16], is popular in forecasting *seasonal* time series. In our settings we assume no seasonality. The smoothing parameters, α and γ, used by the ES method are usually found using grid search.

Note that the above three methods are both *general purpose* (that is, they can fit a large variety of time series) and work *batch* (the models' parameters do not change during the prediction).

An online version of the $ARMA$ model, denoted $O\text{-}ARMA$ was recently analyzed in [1]. The proposed version uses the $ARMA$ model specified in Equation 2, yet the model's parameters may change over time. Nevertheless, note that this method is still general purpose, as is the basic $ARMA$ model.

In this work we provide a solution to the task of *online* predicting the continuation of an assumed *exponential decay* time series. To the best of our knowledge, no intelligent system or machine learning algorithm has addressed this challenged to date.

The above four models (AR, $ARMA$, ES and $O\text{-}ARMA$) are evaluated as baseline models in Section 4 of this study, showing the Exponentron algorithm's superiority.

3 The Exponentron Algorithm

We assume the following set of hypotheses:

$$\hat{s}_t(\boldsymbol{\theta}) = a + b\, e^{-c\,(t-t_0)}, \tag{4}$$

where $\boldsymbol{\theta} = (a, b, c)$ is the set of 3 parameters that should be estimated, $\boldsymbol{\theta} \in \mathbb{R}^3_+$, and $t_0 \in \mathbb{R}_+$ is a time offset parameter.

In this work we focus on the online settings, where learning takes place in rounds. In round t, the algorithm observes the series $(s_0, s_1, \ldots, s_{t-1})$ and is required to make a prediction for the next element in the series, $\hat{s}_t$. The algorithm maintains a set of parameters that are updated every round. After making its prediction, $\hat{s}_t$, the correct value, s_t, is revealed and an instantaneous loss $\ell(\hat{s}_t, s_t)$ is encountered. The round ends with an update of the parameters $\boldsymbol{\theta}$ according to the encountered loss. In this work we use the squared loss function, namely

$$\ell(\hat{s}_t(\boldsymbol{\theta}), s_t) = (\hat{s}_t(\boldsymbol{\theta}) - s_t)^2 = (a + b\, e^{-c\,(t-t_0)} - s_t)^2. \tag{5}$$

Our algorithm, which is called Exponentron, is given in Algorithm 1. The algorithm is aimed at minimizing the cumulative loss.

The algorithm starts with a set of feasible parameters $\boldsymbol{\theta}_0 \in \mathbb{R}^3_+$, that is, $\boldsymbol{\theta}_0 = (a_0, b_0, c_0)$ satisfies the constraints on the parameters. We initialize t_0 by setting the first prediction to be correct, namely, $\hat{s}_t = s_t$ for $t = 0$, and get

$$t_0 = \log((s_0 - a)/b)/c. \tag{6}$$

Now the set of parameters needs to satisfy the constraints $a \leq s_0$, $b \geq 0$ and $c \geq 0$.

The following proposition states that the hypothesis function is a convex function.

Proposition 1. The hypothesis function

$$\hat{s}_t(\boldsymbol{\theta}) = a + b\, e^{-c\,(t-t_0)} \tag{7}$$

is a convex function in the set of parameters $\boldsymbol{\theta} = (a, b, c)$.

Proof. Since $b > 0$ we can rewrite it as $b = e^{\tilde{b}}$, and the hypothesis becomes $\hat{s}_t(\boldsymbol{\theta}) = a + e^{\tilde{b} - c\,(t-t_0)}$. Now it is easy to verify that

$$\hat{s}_t(\alpha\boldsymbol{\theta}_1 + (1-\alpha)\boldsymbol{\theta}_2) \leq \alpha\hat{s}_t(\boldsymbol{\theta}_1) + (1-\alpha)\hat{s}_t(\boldsymbol{\theta}_2),$$

for the sets $\boldsymbol{\theta}_1 = (a_1, \tilde{b}_1, c_1)$, and $\boldsymbol{\theta}_2 = (a_2, \tilde{b}_2, c_2)$ which satisfy the constraints, and $\alpha \in (0, 1)$. □

Since our loss function in Eq. (5) is also a convex function, it turns out that the loss is convex.

Our algorithm is based on *gradient projected methods* [7, pp. 228]. The algorithm starts with a set of feasible parameters $\boldsymbol{\theta}_0 \in \mathbb{R}^3_+$, that is, $\boldsymbol{\theta}_0$ satisfies the constraints. At the t-th round the algorithm predicts the next element in the time series based on the parameters $\boldsymbol{\theta}_{t-1}$. Then, if the encountered loss, $\ell(\hat{s}_t(\boldsymbol{\theta}_{t-1}), s_t)$, is greater than zero, the parameters are updated by a gradient step: $\boldsymbol{\theta}' = \boldsymbol{\theta}_{t-1} + \eta_t \nabla_{\boldsymbol{\theta}} \ell$, where the gradient of the loss is the following vector:

$$\nabla_t = 2(\hat{s}_t(\boldsymbol{\theta}) - s_t)[1, e^{-c\,(t-t_0)}, -b(t-t_0)e^{-c\,(t-t_0)}]. \quad (8)$$

The parameter η_t is the learning rate. Specifically in our case we set $\eta_t = \eta_0/\sqrt{t}$, where η_0 is chosen when the algorithm starts (as in [33, 8]).

At the end of each round the algorithm projects $\boldsymbol{\theta}'$ on a set of constraints in order to get $\boldsymbol{\theta}_t$, a feasible vector.

Algorithm 1 The Exponentron Algorithm

Require: initialize $\boldsymbol{\theta}_0$, learning parameter η.
1: observe s_0 and set t_0 according to Eq. (6)
2: **for** $t = 1, 2, \ldots, T$ **do**
3: predict $\hat{s}_t = a_{t-1} + b_{t-1}\, e^{-c_{t-1}\,(t-t_0)}$
4: observe true value s_t
5: encounter loss $\ell(\hat{s}_t, s_t)$
6: update parameters and project
7: $a_t = \min\{s_0, a_{t-1} - 2\eta_t(\hat{s}_t - s_t)\}$
8: $b_t = \max\{0, b_{t-1} - 2\eta_t(\hat{s}_t - s_t)e^{-c_{t-1}\,(t-t_0)}\}$
9: $c_t = \max\{0, c_{t-1} + 2\eta_t(\hat{s}_t - s_t)b_{t-1}$
$(t-t_0)e^{-c_{t-1}\,(t-t_0)}\}$

Note that the Exponentron algorithm does not require any training before deployment.

Often the performance of an online algorithm is measured by how *competitive* it is with the hypothesis of the best *fixed* parameters $\boldsymbol{\theta}^*$. This is captured by the notion of the algorithm's *regret*, which is defined as the excess loss for not consistently predicting with the parameters $\boldsymbol{\theta}^*$,

$$\text{regret}(\boldsymbol{\theta}^*, T) \triangleq \sum_{t=1}^{T} \ell(\hat{s}_{t-1}(\boldsymbol{\theta}), s_t) - \sum_{t=1}^{T} \ell(\hat{s}_t(\boldsymbol{\theta}^*), s_t). \quad (9)$$

The following theorem states that the regret of the Exponentron algorithm is bounded.

Theorem 2. *The Exponentron algorithm has the following regret bound for every $\boldsymbol{\theta}^*$ in $\mathbb{R}^3_+$,*

$$\text{regret}(\boldsymbol{\theta}^*, T) \leq \frac{\sqrt{T}}{2}\|\boldsymbol{\theta}\|^2 + \frac{1}{2\sqrt{T}} \sum_{t=1}^{T} \|\nabla_t\|^2. \quad (10)$$

Proof. The analysis is based on the stochastic gradient descent with projection analysis [7]. Denote by $\boldsymbol{\theta}_{t-1}$ the set of parameters before the update, by $\boldsymbol{\theta}_{t-1/2}$ the set of parameters after the gradient step, and by $\boldsymbol{\theta}_t$ the set of parameters after the projection step. We have

$$\begin{aligned}
&\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|^2 - \|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|^2 \\
&\quad = \|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|^2 - \|\boldsymbol{\theta}_{t-1/2} - \boldsymbol{\theta}^*\|^2 \\
&\qquad\qquad + \|\boldsymbol{\theta}_{t-1/2} - \boldsymbol{\theta}^*\|^2 - \|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|^2 \\
&\quad \leq \|\boldsymbol{\theta}_{t-1/2} - \boldsymbol{\theta}^*\|^2 - \|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|^2 \\
&\quad = \|\boldsymbol{\theta}_{t-1} - \eta\nabla_t - \boldsymbol{\theta}^*\|^2 - \|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|^2 \\
&\quad = -2\eta\,(\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*) \cdot \nabla_t + \eta^2\|\nabla_t\|^2 \\
&\quad \leq -2\eta\Big(\ell(\hat{s}_t(\boldsymbol{\theta}_{t-1}), s_t) - \ell(\hat{s}_t(\boldsymbol{\theta}^*), s_t)\Big) + \eta^2\|\nabla_t\|^2
\end{aligned}$$

From the second line to the third line we used the property of projections, $\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|^2 \leq \|\boldsymbol{\theta}_{t-1/2} - \boldsymbol{\theta}^*\|^2$. We now sum over all t, and get

$$\begin{aligned}
&\|\boldsymbol{\theta}_T - \boldsymbol{\theta}^*\|^2 - \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|^2 \\
&\leq -2\eta \sum_{t=1}^{T} \Big(\ell(\hat{s}_t(\boldsymbol{\theta}_{t-1}), s_t) - \ell(\hat{s}_t(\boldsymbol{\theta}^*), s_t)\Big) + \frac{1}{2}\eta \sum_{t=1}^{T} \|\nabla_t\|^2
\end{aligned}$$

By rearranging terms we get the desired result. □

4 Exponentron Evaluation

We first evaluate the Exponentron algorithm using synthetic exponential decay time series. Then, we evaluate the Exponentron algorithm in two real-world prediction tasks of significant importance: First, we evaluate the Exponentron in predicting drivers' desired interior cabin temperature during a drive. Specifically, we focus on the cooling condition, where a driver wishes to cool the interior cabin temperature of a car in order to achieve a comfortable state. Second, we wish to predict the number of arriving calls at a call center. Specifically, we consider a call center in which human service agents can handle both inbound calls and other back-office tasks, making the prediction of arriving calls an important factor in real-time work schedule adjustment and managerial decision-making [14].

We compare Exponentron against 4 time series prediction methods, AR, $ARMA$, ES and $O\text{-}ARMA$ which are described in Section 2.

4.1 Synthetic Data Prediction

To gain intuition about the relative strengths of the Exponentron algorithm, we evaluate the Exponentron in a synthetic exponential decay time series prediction task.

To this end, we synthetically generated 1,000 exponential decay time series, which all start at the value 100 at $t = 0$, denoted $s_0 = 100$. Each time series is represented as a tuple $\boldsymbol{\theta} = (a, b, c)$ and is generated according to Equations 4 and 6; $\boldsymbol{s}_t(\boldsymbol{\theta}) = a + b\, e^{-c\,(t-t_0)}$ where $t_0 = \log((s_0 - a)/b)/c$. We synthetically generated the time series by sampling $a \in U[10, 80]$, $b \in U[20, 90]$ and $c \in U[0.1, 0.5]$.[5] We then randomly assigned 900 time series to a training set and the remaining 100 time series were assigned to the test set. The training set time series are used to train the Exponentron algorithm, alongside the four baseline models described in Section 2. The coefficients used by the AR and $ARMA$ methods were learned using a simple linear regression over the training set time series.

[5] a, b and c were chosen as such to allow a significant range of possible hypotheses and yet restrict the range to allow a reasonable first estimation of a, b and c by each of the tested algorithms.

Similarly, the smoothing parameters used by the ES method were found using a grid search. $O\text{-}ARMA$ was implemented according to [1]. The Exponentron's initial parameters, $\boldsymbol{\theta}_0$, were set to the least squares regression parameters calculated based on the training data as described in [32].

The prediction models were tested over the test set time series. Overall, Exponentron significantly outperforms each of the tested baseline models using univariate ANOVA with the prediction method as the independent factor and the prediction mean absolute error as the dependent variable, $p < 0.05$. An illustration of the predicted values made by the Exponentron and the $ARMA$ model is presented in Figure 1.

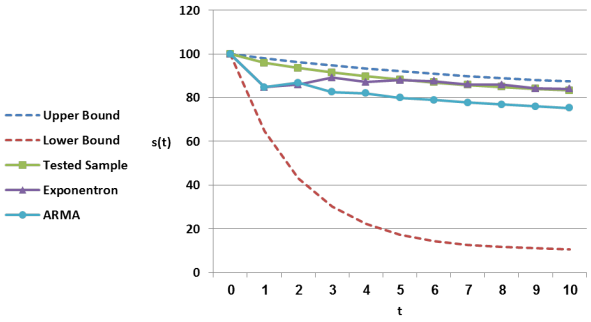

Figure 1: All time series were sampled from the confined space between the upper and lower bounds. The green line is one of the tested time series (represented by $a = 79, b = 20, c = 0.15$) and the purple and light blue lines are the predictions made by the Exponentron and ARMA, respectively.

This synthetic experiment demonstrates the advantage of the Exponentron in exponential decay time series prediction. We now turn to investigate the Exponentron's advantages using two real-world data sets.

4.2 Predicting Desired Climate Changes in a Car's Interior

4.2.1 Data Collection

The cabin temperature time series is $(s_0, s_1, \ldots, s_T)$ where s_0 is the initial cabin temperature when a driver turns the CCS on, and s_j is the cabin temperature k seconds after s_{j-1}. In our setting, k was set to 15 seconds.[6] A driver's *preferred* cabin temperature time series is $(s_0^*, s_1^*, \ldots)$ where $s_0^* = s_0$ and s_i^* is the desired cabin temperature at time frame i. A cabin temperature is said to be **steady** if in a period of 1 minute the driver does not change the CCS features and the cabin temperature does not change more than ϵ. In our setting, ϵ was set to $0.1^\circ\mathrm{C}$.[7] We focus on the task of predicting s_i^* given $(s_0^*, \ldots, s_{i-1}^*)$ until a steady cabin temperature has been reached.

We recruited 28 drivers, ranging in age from 25 to 57 (average of 35), 22 males and 6 females. Each subject was asked to enter a car, which was parked in a garage, in order to experience the environmental conditions – temperatures ranging from $21^\circ\mathrm{C}$ to $31^\circ\mathrm{C}$, averaging $27^\circ\mathrm{C}$. Each subject was presented with a newly designed graphical interface presented on a tablet which we call the *natural interface*. The natural interface presents natural terms such as "Too cold", "Too hot" and "Noisy", which are unavailable in most CCSs. Each subject was instructed to interact with the system, such that after any button was pressed the subject had to *manually* change the features of the CCS using the car's standard interface with the help of our research assistant. Namely, the natural interface did not have any functionality behind it at this point. The session stopped once the cabin temperature of the car was *steady*. While in the car the subject was given a cell phone with a driving simulator "Bus Simulator 3D"[8] to be played while the experiment was conducted. The motivation was to set the conditions similar to regular driving conditions and give the subjects something to do. Unfortunately, due to insurance reasons, we could not conduct the study while subjects were actually driving.[9] The subject was then asked to exit the car for a period of 10 minutes while the car's doors were left open in order to simulate the initial conditions. Note that the subject could choose to click on any button at any given moment, thereby changing the CCS (manually).

During the session, the internal cabin temperature of the car was recorded using a state-of-the-art thermometer that we placed between the driver's and the front passenger's seats. The temperature was measured once every 15 seconds (again, to allow sufficient time for our thermometer to adapt).

Overall, 56 time series were collected. The shortest time series consisted of 6 data points whereas the longest consisted of 26 data points (mean of 13).

4.2.2 Analysis

The Exponentron algorithm, alongside the four baseline models described in Section 2, was evaluated based on the collected data.

The baseline models were trained and evaluated using a one-left-out methodology. Namely, we took out one series at a time from the data set and used the remaining series as training data. All four models were trained as described in Section 4.1.

Note that the Exponentron algorithm and $O\text{-}ARMA$ do not necessitate any training prior to deployment, but instead require an initialization of the parameters (θ_0 in Algorithm 1). Nevertheless, we chose to set the initial parameters to the least squares regression parameters calculated based on the training data using a one-left-out methodology as described in [32]. We also examined the initialization of the Exponentron's parameters using only a subset of the training data. Surprisingly, using *any* single time series from the training data to determine the Exponentron's initial parameters resulted in a less than 10% decrease in the Exponentron's accuracy compared to using all of the training data.

The Exponentron's mean absolute error was found to be $0.13^\circ\mathrm{C}$ per value in the time series. The Exponentron's predictions are 28% more accurate as compared to the best tested baseline model, $O\text{-}ARMA$, which yields a $0.18^\circ\mathrm{C}$ mean absolute error. Table 1 provides a summary of the tested models' prediction errors.

Overall, Exponentron significantly outperforms each of the tested baseline models using univariate ANOVA with the prediction method as the independent factor and the prediction mean absolute error as the dependent variable, $p < 0.05$.

[6] A time interval of 15 seconds was chosen to allow sufficient time for our thermometer to adapt to the changing temperature inside the car. The thermometer specification states that it takes up to 15 seconds for the thermometer to adapt to its environment.

[7] The inaccuracy interval of our thermometer.

[8] Available free at Google Play store.

[9] "Bus Simulator 3D" was also used in previous human-CCS interaction studies in order to simulate driving conditions [5].

Method	Mean Absolute Error (per 15 seconds)
$AR(1)$	0.21
$ARMA(1,1)$	0.2
$ES(1,0)$	0.24
$O\text{-}ARMA(1,1)$	0.18
Exponentron	0.13

Table 1: Prediction of the desired interior temperature of the car. Numbers indicate the mean absolute error made by each prediction method per 15 second frame.

4.3 Inbound Calls in a Real-World Call Center

4.3.1 Real-World Call Center – Secondary Data

We use data that was collected and analyzed in [21]. The data accounts for all inbound calls arriving at the small call center of a bank in 1999 [17]. On weekdays the center is open from 7am to midnight (17 hours) and provides service to over 2,000 callers (on average). We focus on the 16:00-24:00 (8 hours) time frame, in which an average of approximately 750 calls arrive at the call center in an assumed exponential decay manner.

Following the original analysis procedure, we processed the data such that all national holidays were removed and each of the daily recordings was translated into a time series. For this evaluation we used a time series of the form $(c_{17}, c_{18}, \ldots, c_{24})$ where c_i is the number of arriving calls during the $(i-1)^{\text{th}}$ hour of the day. For example, all calls arriving between 17:00 and 18:00 will count as c_{18}.

Overall, 222 time series were constructed. Each time series consists of 8 data points.

4.3.2 Analysis

The Exponentron algorithm, alongside the baseline models described in Section 2, was evaluated using the same procedure as described in Section 4.2.2. Again, we noticed that using *any* single time series from the training data to determine the Exponentron's initial parameters resulted in a less than 15% decrease in the Exponentron's accuracy compared to using the entire training data.

At each time frame of one hour the Exponentron's mean absolute error was found to be 5.8 calls. The Exponentron's predictions are 41% more accurate than the best tested baseline model, $ARMA$, which yields a mean absolute error of 9.8 calls. Table 2 provides a summary of the tested models' prediction errors.

Method	Mean Absolute Error (hourly)
$AR(1)$	10.2
$ARMA(1,1)$	9.8
$ES(0.9,1)$	13.7
$O\text{-}ARMA(1,1)$	9.9
Exponentron	5.8

Table 2: Prediction of call arrivals in a real-world call center. Numbers indicate the mean absolute error made by each prediction method.

Figure 2 demonstrates the predictions provided by the Exponentron and $ARMA(1,1)$ for the number of arriving calls per hour during the evening of February 8^{th}, 1999.

Overall, Exponentron significantly outperforms each of the tested baseline models using pairwise t-tests ($p < 0.05$).

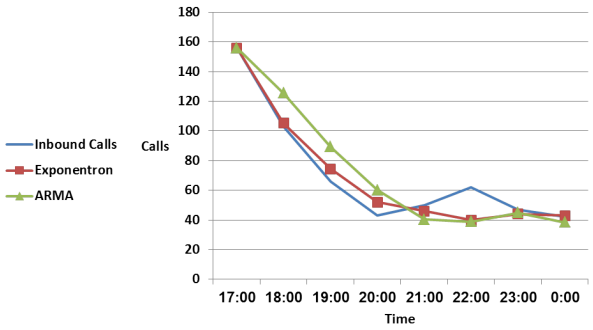

Figure 2: The prediction of inbound calls made by the $ARMA$ and Exponentron models for the evening of 8/2/1999.

5 The Exponentron-Based *NICE* Agent

5.1 The Agent-Human Interaction Challenge

In this section we focus on an agent-human interaction challenge in Climate Control Systems (CCSs). We aim to automatically adjust the CCS features (e.g, fan speed) in order to provide comfortable settings for the user. Specifically, we address a situation in which a driver enters a hot vehicle and the agent's goal is to automatically set and adjust the car's CCS features throughout the ride to the driver's satisfaction.

The agent's main goal is to bring about the driver's desired cabin setting in the car, namely, the appropriate interior cabin temperature and other CCS features. Reaching the target cabin settings is not instantaneous, and may take time and adjustment of the CCS features.

In an interview-based preliminary experiment we conducted with 18 drivers, we noticed that the drivers' satisfaction from their CCS is affected not only by the target cabin setting of the car but, and even more importantly, by the cabin setting endured during the adjustment process. Furthermore, we observed that people have different preferences for both of the above. However, their preferences during the adjustment process exhibit similar *exponential decay tendencies*. For example, Alice's target interior cabin temperature is 21°C and she wishes to reach it as fast as possible (she does not mind enduring extreme CCS settings in the process). Bob, on the other hand, wants to reach 19°C but refrains from settings in which the fan speed is higher than 3 and thus prefers milder adjustments. However, both prefer that the interior temperature decrease exponentially.

5.2 Background

Recent evidence suggests that drivers' current user experience often does not meet drivers' wishes, making many drivers desire more natural car interfaces [19, 27]. For that purpose, some intelligent systems use drivers' observed behavior to automatically elicit the drivers' state or goals [11]. For example, in [29, 30], the authors have shown that learning drivers' behavior can improve the performance of the adaptive cruise control system to drivers' satisfaction. Others offer more expressive interfaces that are more natural for the driver to use and understand [18, 26]. The most relevant works within the context of CCS are [5, 28, 4], in which the authors try to elicit drivers' climate control preferences in order to provide advice to the driver that will help him reduce the climate system's energy consumption. The authors did not account for the possibility of the agent automatically changing the CCS settings nor did they allow for natural input from the driver.

The thermal comfort of human subjects has been exhaustively investigated over the last four decades, resulting in the ISO 7730 standard[10] [2]. The standard, which was also found to be applicable in car cabins, is aimed at predicting the degree of thermal comfort of an *average person* exposed to a certain *steady* environment (see [10] for a recent survey). Unfortunately, the standard does not provide answers on how a system should bring about a comfortable state.

Furthermore, the standard relies on the assumption that user-specific parameters are available such as thermal sensitivity, clothing and activity level. Despite recent attempts to personalize thermal comfort models [3], state-of-the-art thermal comfort models do not provide personalized or adaptive thermal comfort predictions.

Using the Exponentron algorithm, we will next describe a competing approach which does not necessitate the identification of user-specific characteristics prior to its deployment.

5.3 The NICE Agent Design

The NICE agent's goal is to minimize the number of interactions needed by a driver to reach her desired comfort state and maximize the driver's satisfaction from the interaction process.

The agent implements the Exponentron algorithm (Algorithm 1) in order to predict the driver's desired climate changes during the ride and thereby change the CCS setting.

During the process, the driver may provide feedback to the agent using natural comments, such as "Too cold" or "Too hot", using the natural interface described in Section 4.2. These comments, in turn, are used to adapt the Exponentron's predictions, as we will soon describe.

A CCS setting is a tuple $\omega =< temp, f, d >$, where $temp$ is the set temperature (an integer between 16 and 35 degrees C), f is the fan strength (an integer between 1 and 8) and d is the air delivery (1=face only, 2=face and feet, 3=feet). Two additional parameters are e, which is the external temperature (the temperature outside the car), and i, which is the internal cabin temperature. At time t, we denoted the CCS setting as ω_t, the external temperature as e_t and the internal cabin temperature as i_t.

The NICE agent uses 3 models; a *CCS model*, a *human driver model* and an *Exponentron prediction model*. The construction of these models is described later in this section. The NICE agent uses the three models in the following manner: At time t, the NICE agent predicts the driver's desired cabin temperature for the next time frame, $\hat{i}_{t+1}$, using the Exponentron prediction model. Given $\hat{i}_{t+1}$, the agent calls the CCS model and receives and implements a CCS setting ω_t which is predicted to bring about $\hat{i}_{t+1}$. Given that no comment is presented by the driver during the next 15 seconds, the agent assumes that the Exponentron's prediction, $\hat{i}_{t+1}$, and the CCS setting ω_t suit the driver's preferences and the process is repeated. The com signal takes the value of 0 since no comment was given by the driver. The driver can interrupt the above process (which otherwise will continue throughout the entire ride) by providing feedback. If a comment (c) is given within 15 seconds of implementing ω_t, then the agent uses the human driver model to predict the driver's desired CCS setting, $\hat{\omega}_t$, and implements it instantaneously in the CCS. Then, the system maintains the new CCS setting until 15 seconds pass in which no further feedback is provided by the driver. Namely, if the driver provides another comment within 15 seconds of his last comment, the human driver's model is called on once again and the 15-second timer is re-set. Once 15 seconds pass without further comments, the resulting cabin temperature is used to update the Exponentron's parameters. To that end, the com signal is set to 1. That is, the Exponentron's parameters can only be adjusted when the driver interacts with the agent. Figure 3 illustrates the agent's algorithmic scheme.

[10] Also known as Fanger's Predicted Mean Vote (PMV) criteria.

5.3.1 The CCS Model

Recall that the CCS model is used to determine which CCS setting ω_t will bring about the desired change in the internal cabin temperature over a course of 15 seconds. For that purpose, the model receives i_t, ω_{t-1} and $\hat{i}_{t+1}$.

In order to train the CCS model, thirty distinct CCS settings were selected such that their set temperature, $temp$, was lower than the initial cabin temperature, i_0, at the time of the experiment. This property is required to enforce a cooling condition, which we examine in this study. We counter-balanced the selected CCS settings to account for the different possible ωs; namely, different $temp$, f and d values. Each CCS setting was manually configured to the CCS at the beginning of the trial. The cabin temperature, i, and the external temperature, e, were recorded every 15 seconds until the car's cabin temperature reached a steady state. Between every 2 consecutive experiments the car was turned off and the car's doors were left open for 10 minutes so as to simulate the initial conditions.

From the 30 trials we conducted over the course of 3 days, we recorded 657 measurements. Each of the measurements corresponds to a change in the car's cabin temperature, $i_{j+1} - i_j$, given e_j, and the CCS setting ω used in the trial. We fit the data using a simple linear regression model which yields the best fit out of the tested models[11]. Namely, we constructed a model which, given i_t and ω, predicts i_{t+1}. To find a ω which is most likely to bring about the desired change, we iterate through all possible ωs. In the case of a tie, where more than a single CCS setting is expected to change the cabin temperature in the desired manner, the model outputs one of the CCS settings which is most similar to the previous CCS setting, ω_{t-1}. Recall that a CCS setting is a vector $< temp, f, d >$, therefore similarity is easily defined. In this work we used the cosine similarity.

Using cross-validation, the learned model yields a mean absolute error of $0.15^{\circ}\mathrm{C}$ and a strong correlation coefficient of 0.9. In comparison, using the last cabin temperature change, $\Delta_j = i_j - i_{j-1}$, as an estimation for the next cabin temperature, $\hat{i}_{j+1} = i_j + \Delta_j$, yields a mean absolute error of 0.51 and a correlation coefficient of 0.3.

5.3.2 The Human Driver Model

Given a driver's comment, denoted c_0, the human driver model is used to predict the driver's desired CCS settings. The model is based on multi-dimensional regression: At time t when a comment (denoted as c) is given (i.e, a button is pressed), the model predicts the desired CCS setting $\hat{\omega}_t$, given ω_t, e_t, i_t, c_0 and the last 2 previously provided comments, denoted c_1, c_2.

In order to train the human driver model, we used the data collected in Section 4.2. Recall that in our experiment drivers were asked to interact with the newly designed natural interface while changing the CCS settings *manually*. The experiment recordings were translated into more than 100 vectors of the form $<$

[11] We also examined other, more sophisticated modeling, for example using SVM with kernels. These models did not provide a significant improvement in prediction accuracy.

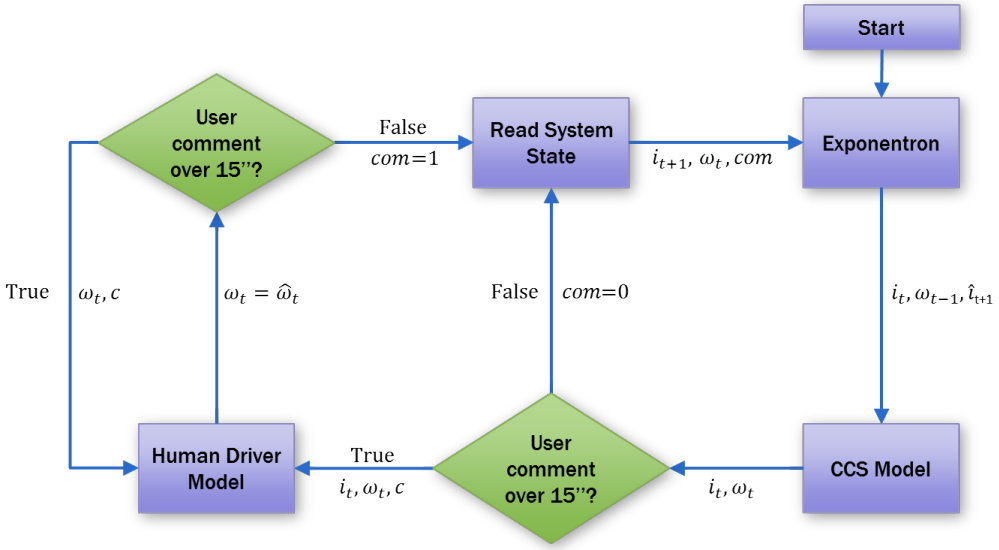

Figure 3: The NICE agent's algorithmic scheme.

$\omega_t, e_t, i_t, c_0, c_1, c_2 >$ as described above, with the drivers' manually set CCS setting, ω', as their label. Each session resulted in a different number of vectors, depending on the session's length.

The multi-dimensional regression consists of 3 linear regression models, each predicting a different component of the desired CCS setting $\omega' =< temp, f, d >$. Using cross-validation, the prediction model yields a mean absolute error of 0.9 in predicting the next fan speed, f, and a mean absolute error of 1.02°C for the next set temperature, t. A high 97% accuracy in predicting the desired air delivery, d, was also recorded.

5.3.3 *The Exponentron Prediction Model*

The Exponentron prediction model implements the Exponentron algorithm (Algorithm 1). The Exponentron is trained with the same procedure used in Section 4.2. The model receives an additional input bit *com*, signaling whether the driver provided a comment in the last time frame. If and only if the *com* bit is 1, then an adaptation of the model parameters is executed using the current cabin temperature i_t.

The θ learned parameters represent the driver's preferences. Specifically, the parameter a represents the driver's intended steady state cabin temperature and the parameters b and c represent the way in which the driver wishes to bring about the desired cabin temperature.

5.4 Evaluation

5.4.1 *Experimental Methodology*

We recruited 24 drivers who did not participate in the data collection phase described in Section 4.2, with an equal number of males and females, ranging in age from 25 to 60 (average of 34). In a similar protocol to that described in Section 4.2, each subject was asked to enter a car that was parked in a garage, recreating the environmental conditions with temperatures ranging from 32°C to 37°C, averaging 35°C. Each subject participated in two consecutive trials. In each trial the subject was equipped with either the *Technical CCS* or the *NICE agent*. The technical CCS presented buttons similar to those available in the common CCS, with which the driver can explicitly select her desired CCS setting. Namely, it presented two scales: one for the fan speed and the other for the temperature. The driver could change the setting by selecting her preferred fan speed and temperature on the designated scales. Namely, in a single interaction, the driver could change the CCS setting completely. Note that no intelligent agent was implemented to support it. The NICE agent uses the natural interface which is the same interface as described in Section 4.2. In both conditions, the GUI was presented on a tablet covering the car's central stack in order to avoid biasing the results. Each subject was instructed to interact with the system as she saw fit by using the buttons available in the presented interface. While in the car the subject was given a cell phone with a driving simulator "Bus Simulator 3D" to be played while the experiment was conducted.

Once the cabin temperature, i, reached a steady state, the session came to an end. Each session lasted 2-6 minutes (mean of 4 minutes). After the session ended, the driver was asked to exit the car for a period of 10 minutes while the car's doors were left open in order to simulate initial conditions. The process was repeated once more under the condition that was not examined in the first session. Subjects were counter-balanced as to which condition they experienced first in order to maintain the scientific integrity of the results.

During each session we recorded the number of interactions needed by the driver in order to reach her desired steady state. At the end of the experiment, drivers were asked to fill out a post-experiment questionnaire aimed at evaluating their satisfaction from the examined interfaces.

5.4.2 *Results and Analysis*

We first analyze the number of interactions needed by drivers to reach their desired steady states under the examined conditions. Then we summarize the subjects' answers in the post-experiment questionnaire. Note that the technical CCS is the current state-of-the-art CCS and acts as the benchmark in the following analysis.

The NICE agent required a significantly lower number of interactions from the driver compared to the technical CCS using t-test ($p < 0.05$). The NICE agent averaged 5.35 interactions carried out

by the driver until a steady state was reached while the technical CCS averaged 6.54 interactions. Out of the 24 subjects, only 8 subjects required more interactions while equipped with the NICE agent compared to their benchmark score. See Figure 4 for a summary.

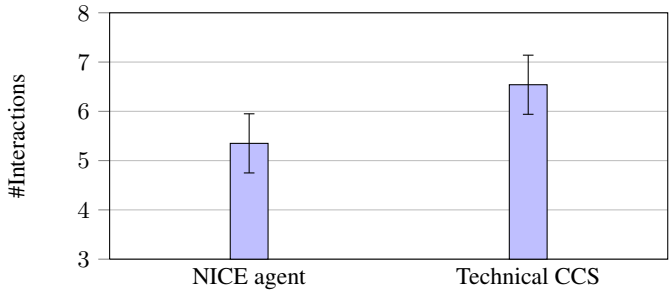

Figure 4: Average number of interactions per interface; (the lower the better). Error bars indicate standard errors.

Recall that the NICE agent's goal is to automatically set and adjust the car's CCS setting throughout the ride to the driver's satisfaction. In order to assess the driver's satisfaction from the interaction, at the end of the experiment we asked each driver which, if any, of the tested conditions she would want to see available in her car. Out of the 24 subjects, 13 subjects stated that they want to use the NICE agent, while 10 subjects stated their preference for the technical CCS. We also asked subjects to state their satisfaction level from the tested conditions. Subjects reported an average score of 6.2 out of 10 when asked for their satisfaction from the technical CCS. This result is significantly lower, using t-test ($p < 0.05$), compared to the NICE agent which recorded an average score of 7.2 out of 10.

To summarize, the results indicate that the NICE agent is able to reduce the number of interactions needed by drivers in order to achieve their desired comfort states (6.54 vs. 5.35) to the drivers' satisfaction, as portrayed in the increase of the subjects' subjective satisfaction (7.2 vs. 6.2) and the subjects' preferred interaction mode (13 vs. 10).

6 Conclusions

In this paper we presented the Exponentron algorithm for the online prediction of assumed exponential decay time series. The Exponentron algorithm was evaluated both theoretically and empirically; theoretically we show a regret bound that compares our algorithm to the best batch algorithm, which is given the entire time series in advance. Empirically, the Exponentron was evaluated in synthetic and real-world prediction tasks in which it significantly outperformed classic time series prediction methods. Furthermore, we demonstrated the Exponentron algorithm's benefit using the novel NICE agent, which significantly enhances the driver-automotive CCS interaction process compared to standard CCS control.

From an applicative perspective, our proposed methodology is not restricted to CCS-based agents. For example, in the development of personal assistance agents, the prediction of human forgetfulness may be beneficial. Specifically, an agent may need to predict the time it would take for its user to forget an important piece of information and provide a reminder for it. The forgetting curve [13] predicts the decline of memory retention in time and is assumed to decay exponentially for all people, though significant differences between individuals may be observed. Significant differences over time may also be presented for any specific person, which would necessitate online adaptation. In a similar fashion to the natural interface, which was presented in Section 4.2, a user can express her feedback in a natural manner, e.g., "Remind me later".

Future work will include the investigation of other time-dependent phenomena that are likely to adhere to an a priori assumed functional behavior. For example, we will tackle the challenge of automatically adjusting exercise levels in online tutoring systems. The progress in which new skills are learned is commonly assumed to follow a sigmoid curve, with some measure of skill on the Y axis and the number of trials on the X-axis [23]. In the spirit of the presented work and the NICE agent's design, the agent will be able to adjust, online, the exercise level according to its estimation of the student's learning curve. The student will be able to express her feedback in a natural manner, for example "The exercises are too difficult", and thus make the agent adapt its behavior.

ACKNOWLEDGEMENTS

We would like to thank the ERC (grant #267523) for their support in this research.

REFERENCES

[1] Oren Anava, Elad Hazan, Shie Mannor, and Ohad Shamir, 'Online learning for time series prediction', in *Proc. of the Conference on Learning Theory*, pp. 172–184, (2013).
[2] ASHRAE, 'Standard 55-2013. thermal environmental conditions for human occupancy. ashrae', *American Society of Heating, Refrigerating and Air-Conditioning Engineering*, (2013).
[3] Frederik Auffenberg, Sebastian Stein, and Alex Rogers, 'A personalised thermal comfort model using a bayesian network', in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, (2015).
[4] Amos Azaria, Yaakov Gal, Sarit Kraus, and Claudia V Goldman, 'Strategic advice provision in repeated human-agent interactions', *Autonomous Agents and Multi-Agent Systems*, **30**(1), 4–29, (2016).
[5] Amos Azaria, Ariel Rosenfeld, Sarit Kraus, Claudia V Goldman, and Omer Tsimhoni, 'Advice provision for energy saving in an automobile climate-control system', *AI Magazine*, **36**(3), 61–73, (2015).
[6] José M Bernardo and Adrian FM Smith. Bayesian theory, 2001.
[7] Dimitri P Bertsekas, 'Nonlinear programming', (1999).
[8] Léon Bottou, 'Stochastic gradient tricks', *Neural Networks, Tricks of the Trade, Reloaded*, 430–445, (2012).
[9] George Box, Gwilym M. Jenkins, and Gregory C. Reinsel, *Time Series Analysis: Forecasting and Control*, Prentice-Hall, 1994.
[10] Cristiana Croitoru, Ilinca Nastase, Florin Bode, Amina Meslem, and Angel Dogeanu, 'Thermal comfort models for indoor spaces and vehiclescurrent capabilities and future perspectives', *Renewable and Sustainable Energy Reviews*, **44**, 304–318, (2015).
[11] Sergio Damiani, Enrica Deregibus, and Luisa Andreone, 'Driver-vehicle interfaces and interaction: where are they going?', *European transport research review*, **1**(2), 87–96, (2009).
[12] Norman Richard Draper, Harry Smith, and Elizabeth Pownell, *Applied regression analysis*, volume 3, Wiley New York, 1966.
[13] Hermann Ebbinghaus, *Memory: A contribution to experimental psychology*, number 3, University Microfilms, 1913.
[14] Noah Gans, Ger Koole, and Avishai Mandelbaum, 'Telephone call centers: Tutorial, review, and research prospects', *Manufacturing & Service Operations Management*, **5**(2), 79–141, (2003).
[15] Everette S Gardner, 'Exponential smoothing: The state of the art', *Journal of forecasting*, **4**(1), 1–28, (1985).
[16] Paul Goodwin, 'The holt-winters approach to exponential smoothing: 50 years old and going strong', *Foresight: The International Journal of Applied Forecasting*, (19), 30–33, (2010).
[17] I. Guedj and A. Mandelbaum, 'Call center data', Technical report, Technion, Israel Institute of Technology, (2000).
[18] Jee Yeon Hwang, Kent Larson, Ryan Chin, and Henry Holtzman, 'Expressive driver-vehicle interface design', in *Proceedings of the 2011 Conference on Designing Pleasurable Products and Interfaces*, p. 19. ACM, (2011).

[19] Li Li, Ding Wen, Nan-Ning Zheng, and Lin-Cheng Shen, 'Cognitive cars: A new frontier for adas research', *Intelligent Transportation Systems, IEEE Transactions on*, **13**(1), 395–407, (2012).

[20] Chenghao Liu, Steven CH Hoi, Peilin Zhao, and Jianling Sun, 'Online arima algorithms for time series prediction', in *Thirtieth AAAI Conference on Artificial Intelligence*, (2016).

[21] Avishay Mandelbaum, Anat Sakov, and Sergey Zeltyn, 'Empirical analysis of a telephone call center', Technical report, Technion, Israel Institute of Technology, (2001).

[22] Luis Moreira-Matias, Joao Gama, Michel Ferreira, João Mendes-Moreira, and Luis Damas, 'Predicting taxi–passenger demand using streaming data', *Intelligent Transportation Systems, IEEE Transactions on*, **14**(3), 1393–1402, (2013).

[23] Jaap MJ Murre, 'S-shaped learning curves', *Psychonomic bulletin & review*, **21**(2), 344–356, (2014).

[24] Isaac Newton, *Scala graduum caloris: calorum descriptiones & signa*, Royal Society of London, 1701.

[25] Eoin O'Mahony and David B Shmoys, 'Data analysis and optimization for (citi) bike sharing', in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, (2015).

[26] Ioannis Politis, Stephen Brewster, and Frank Pollick, 'To beep or not to beep?: Comparing abstract versus language-based multimodal driver displays', in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 3971–3980. ACM, (2015).

[27] Simon Ramm, Joseph Giacomin, Duncan Robertson, and Alessio Malizia, 'A first approach to understanding and measuring naturalness in driver-car interaction', in *Proceedings of the 6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pp. 1–10. ACM, (2014).

[28] Ariel Rosenfeld, Amos Azaria, Sarit Kraus, Claudia V Goldman, and Omer Tsimhoni, 'Adaptive advice in automobile climate control systems', in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 543–551. International Foundation for Autonomous Agents and Multiagent Systems, (2015).

[29] Avi Rosenfeld, Zevi Bareket, Claudia V Goldman, Sarit Kraus, David J LeBlanc, and Omer Tsimhoni, 'Towards adapting cars to their drivers', *AI Magazine*, **33**(4), 46, (2012).

[30] Avi Rosenfeld, Zevi Bareket, Claudia V Goldman, David J LeBlanc, and Omer Tsimhoni, 'Learning drivers behavior to improve adaptive cruise control', *Journal of Intelligent Transportation Systems*, **19**(1), 18–31, (2015).

[31] Avraham Shvartzon, Amos Azaria, Sarit Kraus, Claudia V Goldman, Joachim Meyer, and Omer Tsimhoni, 'Personalized alert agent for optimal user performance', in *Proceedings of the 30th International Conference on Artificial Intelligence (AAAI)*. AAAI, (2016).

[32] Gordon K Smyth, 'Nonlinear regression', *Encyclopedia of environmetrics*, (2002).

[33] Martin Zinkevich, 'Online convex programming and generalized infinitesimal gradient ascent', (2003).