An Ensemble Technique for Better Decisions Based on Data Streams and its Application to Data Privacy

Fabian Laforet, Christian Olms, Rudolf Biczok and Klemens Böhm

Abstract—In this work, we address the problem of making decisions based on data streams, i.e., choosing an action when a new value is recorded. For instance, actions can be trading decisions in financial markets, choices of controllers in dynamic systems or perturbations of the data stream itself. To start with, we propose a language that allows individuals to formulate requirements on the action space. We use prediction techniques to identify the best possible action. However, for many scenarios there is not just one technique that predicts the future precisely, and different techniques behave quite differently. Thus, since there is no technique that dominates all the others, our conclusion is to take multiple predictions generated by different techniques into account. While ensemble techniques aggregating the predictions seem promising, existing techniques have issues, such as unnecessary information losses or the need for a predefined quality measure. Thus, we propose a new ensemble approach that weights predictions techniques according to requirements and solves an optimization problem that derives decisions directly from weighted predictions. We apply our solution to data privacy on data streams. For this setting, the benefits provided by prediction techniques have not been studied yet. In three case studies, we show that our solution consistently achieves better decision-making quality than approaches from related work.

Index Terms—Decision support, Time series analysis, Privacy, Markov processes, Constrained optimization

1 Introduction

Making predictions based on data streams is a data analysis task which has received much attention over the years. Predictions are expectations or probability distributions of future values. Importantly, they are not an end in itself, but support decision making [1]. Making a decision means that an action out of a given action space is chosen. To distinguish between feasible and infeasible or good and bad decisions, one can define requirements (aka. constraints) on the action space. Think of a trader who decides at each point of time at which price to order a stock. Here, a requirement that constrains the action space is that orders must not exceed the liquidity of the trader. Given this requirement, traders now have the objective to buy stocks with the largest gain. Clearly, predictions of future prices help to make decisions to maximize the expected return. This article aims for a framework to make good decisions based on data streams in the presence of requirements on the action space.

Another scenario where predictions help to make decisions is the perturbation of data streams, in order to ensure privacy. This setting is of interest to the database community by itself. To our knowledge, there has not been much research regarding the benefits provided by prediction techniques. This setting will be our running example.

Data Perturbation as running example. *Streaming data may be sensitive with regard to privacy. For instance, healthcare data allows inferences on unhealthy lifestyles. Analyzing energy consumptions allows burglars to identify when residents are at home [2]. Data perturbation shields from such risks. Perturbation of streaming data means that, every time a new value is recorded, a decision-making mechanism decides on a value to replace the original one.*

Research has proposed different perturbation schemes for data streams. We now map these schemes to requirements.

Data-Perturbation Requirements. *Some schemes perturb the data so that certain private information is not inferable. For instance, proposals are the addition of random values to the records [3] that must follow a certain distribution to guarantee differential privacy [4], [5]. Other schemes smooth the perturbed data stream as well as possible [6] to hide private information. We call such characteristics of the perturbation* privacy requirements. *On the other hand, some applications require certain information to remain in the data. An example is that the sum of energy consumptions must not change, to facilitate invoicing [2]. These are what we call* utility requirements. *In the context of energy, there are proposals to deploy batteries whose charging perturbs the actual energy consumed at a certain point of time [7]. Here, a perturbation mechanism must consider any characteristic of the battery such as its capacity when controlling the charging rate. We refer to such restrictions as* technical requirements.

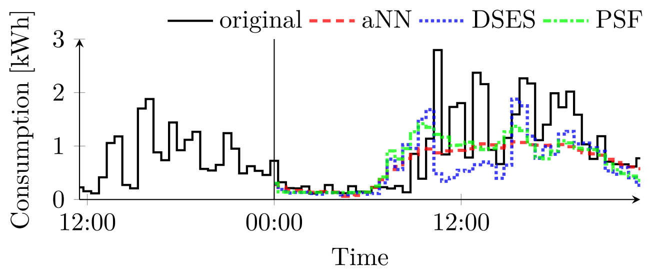
From our running example, we see that there are various requirements one can choose from. In addition, these requirements do not exist in isolation, i.e., one often must combine one with others.

Predictions [8] can support the decision how to perturb records. The following example illustrates this.

Predictions for better perturbations. *Think of the requirement to smooth a data stream as well as possible [6]. The perturbation of the current point of time also affects the smoothness in the future. Thus, predictions can help to make better decisions [9]. In particular, one should not choose*

- *Fabian Laforet and Klemens Böhm are with Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany. E-mail: laforetf@ira.uka.de, klemens.boehm@kit.edu*
- *Christian Olms and Rudolf Biczok are with Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany. E-mail: {christian.olms, rudolf.bizcok}@student.kit.edu*

Manuscript received . . . ; revised . . . .



Comparison to original w.r.t.	aNN	DSES	PSF
Euclidean distance	**4.01**	4.69	4.06
Deviation in overal consumption	12.1	13.7	**9.6**
Range coverage	36.0%	**65.8%**	48.0%

Fig. 1. Prediction example

a perturbation that maximizes the smoothness between the previous and the current record, but one that optimizes the expected smoothness of the entire series, including predictions.

In consequence, we see two research questions, one generic for decision making, and the other one specific for data perturbation: First, how to select prediction techniques for arbitrary requirements automatically, and how to arrive at optimal decisions based on the predictions? I.e., we propose a framework to make decisions based on individually defined requirements and on a set of prediction techniques. Second, given a general solution to this question, how to adapt it to the perturbation scenario? In particular, we formulate perturbation characteristics from related work as requirements. – We now explain the difficulties that come with these questions.

Difficulties. Preliminary experiments of ours confirm that different techniques give way to prediction with different properties, as Example 1 illustrates.

Example 1. *The black continuous line in Figure 1 graphs the energy consumption of a household. To predict the consumption of the next day, we apply three state-of-the-art prediction techniques [8]: artificial Neural Network (aNN), Double Seasonality Exponential Smoothing (DSES) and Pattern Sequence based Forecasting (PSF). All techniques behave differently. To actually choose one, one may want to rely on quality measures. Here, we have applied three measures comparing the original with the predicted series: (1) the Euclidean distance, (2) difference of the overall consumption of the next 24 hours and (3) the range coverage. Figure 1 reveals that these techniques perform quite differently regarding the quality measures.*

In Example 1, we use different quality measures to illustrate the properties of prediction techniques. A core hypothesis is that different prediction techniques are suited for different decision-making problems. This is because quality measures are differing in relevance in different application scenarios, with no measure being superior throughout.

Example 2. *Consider the different properties of the techniques from Example 1. In general, one could apply aNNs since the error is minimal for each point of time. However, if one has the requirement to smooth the data stream, knowing the range of values in the near future is important. Or, if a battery performs the perturbation, and the sum of the perturbations must not violate capacity restrictions, knowing the consumption over several future records is useful. For the first requirement, DSES shows the highest accuracy. For the second requirement, PSF is optimal.*

In general, the connections between the properties of different prediction techniques and the requirements are not clear. I.e., one cannot formulate or fix aprori the required quality measures to evaluate different prediction techniques. In consequence, the first difficulty is to identify the connection *automatically* to select appropriate predictions.

Next, if one formulates several requirements, the selection scheme of prediction techniques must consider different properties. At a first sight, existing ensemble techniques [10] seem appropriate to combine different predictors. However, we cannot readily adapt them for our scenario. This is because they require a quality measure for predictions as input, but the connection between quality measures and requirements is not known. Even if we knew appropriate quality measures, existing ensemble techniques would not be readily applicable: Since those techniques rely on exactly one measure, one would need additional knowledge on how to combine them. Quality measures have different domains and units. In consequence, it is hard to impossible to derive a general rule on how to combine them. Finally, existing ensemble techniques produce a single prediction by aggregating several ones [11]. This might result in a loss of information. In Example 1, combining the prediction of DSES with any other prediction decreases the range coverage. So we aim at making decisions by deriving decisions *directly* from the available predictions.

When it comes to data perturbation, one must identify a language that facilitates the representation of respective requirements. It should cover the full range of privacy, utility and technical requirements from related work. For instance, it will be necessary to verify that the formulation can express and provide differential privacy [12], preserving correct sums of records [2] or simulating the behavior of a battery [7].

Contributions. Our contributions are as follows:

(1) We treat the search for a good perturbation as an optimization problem, which we derive from a set of requirements. We introduce two types of requirements, one to distinguish between feasible and infeasible and another one to distinguish between good and bad decisions. We show that our solution allows to represent privacy, utility and technical requirements.

(2) Next, we demonstrate that existing approaches using predictions to make decisions do not meet our objectives. To do so, we describe frameworks that are generalizations of existing approaches. We refer to them as baselines. (2a) The first baseline is an optimization problem that uses a single prediction to arrive at a decision. We demonstrate that its solutions are suboptimal. (2b) The second baseline applies a Markov Decision Process that considers several possible futures. However, it relies on predictions that result from Markov chains whose optimal design is unknown since it depends on the requirements. In addition, decisions are suboptimal, due to several discretization steps.

(3) We then propose a framework that makes decisions based on streaming data in the presence of requirements, as follows: We formulate decision making as an optimization problem depending on several predictions. Specifically,

since prediction techniques perform differently, we propose an ensemble-weighting scheme. It applies individually defined requirements to evaluate prediction techniques. The framework does not require any individually defined quality measures.

(4) For our evaluation, we focus on perturbations and apply combinations of privacy, utility and technical requirements. Our experiments show that our solution consistently achieves better perturbation compared to the baselines when applying the same prediction techniques.

2 Related Work

We first review approaches that use predictions and ensemble techniques to make decisions on streaming data. We then look at data-perturbation approaches in streaming scenarios.

2.1 Decision Making on Streaming Data

State-of-the-art approaches coming from control theory apply prediction techniques to make good decisions. Many of them rely on single prediction techniques [8], e.g., they predict energy consumption to implement demand-response systems [13] or they predict traffic to anticipate traffic conditions [14]. In some scenarios it is possible to increase the accuracy of predictions to make better decisions, e.g., for wind forecasting, by creating ensembles over different techniques [15]. In addition, methods which adapt ensemble methods for supervised machine-learning tasks from static data to data streams have been proposed over the years [16]. These solutions take multiple predictions and aggregate them to a single prediction. In general, their main challenge when applying existing ensemble mechanisms [17] are concept drifts. This means that the environments are dynamic, so concepts from which the data stream is generated may shift. The range of their solutions covers approaches which replace [18] or update [19] predictors, their parameters [20] or their ensemble-weights [21] over time, eventually by following different rules according to the type of the drift [22]. All this is orthogonal to our work and can be combined with it. When working with data streams that are impossible to predict accurately, just aggregating over predicted values may yield suboptimal results. This is because valuable information provided by each prediction, e.g., the range coverage, is lost when just aggregating the predicted values.

So techniques that predict probability distributions, e.g., Markov chains, are preferred. Markov Decision Processes (MDPs) add decision components to the Markov chain which enable decision making. Recent examples are to decide when to charge vehicles optimally [23] or when to intervene for health events [24]. However, as we illustrate in Section 4.2, MDPs suffer from unknown optimal design of the state space and from reduced decision quality due to discretization.

2.2 Privacy Approaches

In general, data streams contain private information which one can infer. For instance, stalkers can track their victims [25], or companies can perform targeted advertising that may result in price discrimination [2].

One approach to perturb data adds random values to each record [3]. To guarantee that certain information remains secret as defined by differential privacy [12], state-of-the-art approaches let the perturbation follow a certain distribution [4], [26]. Other approaches ensure privacy by smoothing data streams to hide activities [6]. Although those approaches can hide certain information, they tend to leave aside that the perturbed data must fulfill utility requirements.

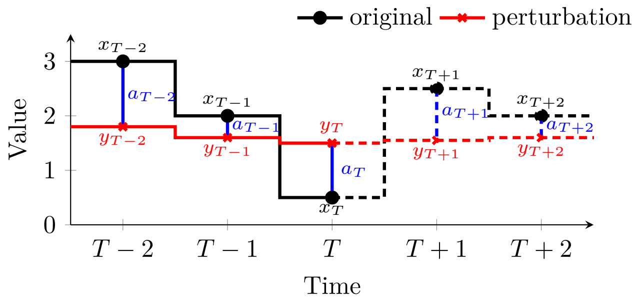


Fig. 2. Illustration of notation for perturbations

Other approaches consider the utility of the data by minimizing the difference between the original and the perturbed data streams [27]. Next to utility requirements, there may be technical restrictions: To cope with legal constraints, current research proposes to use charging rates of batteries to perturb individual energy consumptions [7]. Since the perturbation is limited by the capacity and the power of the battery, a perturbation mechanism must consider these restrictions.

The approaches listed here are specific solutions for a given use case, and their privacy, utility or technical requirements are not combinable in a straightforward way. We apply their characteristics to define requirements to verify the generality of our solution. [28] proposes an approach to combine privacy constraints, but this solution is for static data only. I.e., it does not use predictions and is not applicable to data streams.

3 Fundamentals

In this section, we present the notation and say how requirements on the action space are represented.

3.1 Notation

To formulate requirements on numerical streaming data, we use the following notation: A stream is a sequence of records over time. Individuals record values x_t at points of time t. We call the current point of time T, i.e., the new value is x_T. At each point of time, after observing x_T, an action a_T is chosen. To determine a_T, information on the current and previous records $x_T, x_{T-1}, x_{T-2}, \dots$ and actions $a_{T-1}, a_{T-2}, \dots$ are used, as well as expectations regarding future records $x_{T+1}, x_{T+2}, \dots$ that result from prediction techniques.

Figure 2 illustrates the notation for our perturbation example on streaming data. Here, a_T is the deviation added to x_T to obtain the perturbed value y_T.

3.2 Requirements

We distinguish between two types of requirements individuals may have, namely *strict* and *soft requirements*. Strict requirements define properties each action must fulfill, *soft requirements* define properties actions should fulfill as well as possible, and individuals can select several of them. We now define these types mathematically and show that they cover our related work.

Definition 1 (Strict requirement)**.**
Let $f_{strict}(x_T, x_{T-1}, \dots, a_T, a_{T-1}, \dots)$ be a function that is defined on the current and previous records $x_T, x_{T-1}, \dots$ and actions $a_T, a_{T-1}, \dots$. The inequation

$$f_{strict}(x_T, x_{T-1}, \dots, a_T, a_{T-1}, \dots) \leq 0$$

is a strict requirement.

We refer to f_{strict} as *strict requirement function.* The evaluation of strict requirements results from current and previous points of time only. This is because it is impossible to predict future records precisely. Individuals should specify strict requirements in a way that, independently of what has happened in the past, an action a_T exists that fulfills all of them. In case that no feasible a_T can be identified, one must specify an exception handling such as "do nothing" or "set a_T to a random value". Observe that such exception handling is necessary in general and is not specific to our approach.

We now present the formulation of the perturbation characteristics presented in Section 2.2 as strict requirements.

Example 3. *To provide differential privacy [12], the deviations must follow a certain distribution, e.g., a Binomial one [4], [26]. To formulate this use case, the inequation*

$$f_{strict}^{diff\text{-}priv} = \tau - \alpha(a_T, a_{T-1}, \dots) \leq 0$$

is a strict requirement, where $\alpha(a_T, a_{T-1}, \dots)$ is the level of significance rejecting a statistical test that the deviations follow different distributions, and τ is the minimum level of significance that is acceptable.

Example 4. *[7] proposes to charge batteries so that energy-consumption data is perturbed. In consequence, one must formulate technical restrictions as strict requirements. The maximum amount of energy p_{max} chargeable between two points of time results in the requirement*

$$f_{strict}^{power} = |a_T| - p_{max} \leq 0.$$

In addition, the load level of a battery must be larger than zero and smaller than its capacity c:

$$f_{strict}^{cap1} = -\left|\sum_{t=0}^{T} a_t\right| \leq 0 \ \wedge \ f_{strict}^{cap2} = \left|\sum_{t=0}^{T} a_t\right| - c \leq 0$$

The formulation of the correct sum of multiple records to facilitate correct invoicing [2] is straightforward by defining error bounds that decrease during the invoicing period.

Next, we formulate soft requirements that actions should fulfill as well as possible.

Definition 2 (Soft requirement)**.**
Let $f_{soft}(x_t, x_{t-1}, \dots, a_t, a_{t-1}, \dots)$ be a function that is defined on the current and previous records of $x_t, x_{t-1}, \dots$ and actions $a_t, a_{t-1}, \dots$. The term

$$\min \sum_t f_{soft}(x_t, x_{t-1}, \dots, a_t, a_{t-1}, \dots)$$

is a soft requirement.

We refer to f_{soft} as *soft requirement function.*In contrast to strict requirements that check for fulfillment at each point of time, soft requirements require a minimum at a (finite) time horizon. If one formulates more than one soft requirement, we assume that they all are equally important. At each current point of time T, f_{soft} describes the impact a_T has on the objective function in the absence of future records. However, the instantiation a_T can influence the quality of future actions, e.g., if the data stream is smoothed while future records increase, one should increase the current record to minimize the differences between subsequent records. In consequence, choosing an instantiation of a_T that does not maximize the quality of the current action might be necessary in order to minimize f_{soft} over *all* points of time t.

Now we formulate the requirements from our related work we have not discussed yet using soft requirements.

Example 5. *To hide activities, individuals might want to smooth their data streams [6], i.e., to minimize the difference between subsequent records. A soft requirement that represents this is*

$$\min \sum_t f_{soft}^{diff} = \min \sum_t (x_t + a_t - x_{t-1} - a_{t-1})^2$$

Example 6. *An individual might want to reduce the differences between the original and the perturbed records [27]. A respective soft requirement can be the quadratic distance:*

$$\min \sum_t f_{soft}^{error} = \min \sum_t a_t^2$$

3.3 Optimal Decisions

Having requirements, the question remains how to arrive at a decision, i.e., choosing good actions. Under the condition that all strict requirements are fulfilled, an optimal action sequence $a_0, \dots, a_\Omega$ fulfills the average of soft requirements as well as possible, where Ω is the time horizon. I.e., an action sequence that fulfills all strict requirement is optimal if there is no other action sequence that also fulfills all strict requirements but results in a better (smaller) value w.r.t. soft requirements.

Definition 3 (Optimal action sequence)**.**
The quality of the action sequence is optimal iff

$$\begin{aligned}
\forall T \in [0, \Omega], \forall f_{strict}: \ & f_{strict}(x_T, x_{T-1}, \dots, a_T, a_{T-1}, \dots) \leq 0 \\
& AND \\
\nexists a'_0, a'_1, \dots, a'_\Omega: \ & \sum_{f_{soft}} \sum_t f_{soft}(x_t, x_{t-1}, \dots, a'_t, a'_{t-1}, \dots) < \\
& \sum_{f_{soft}} \sum_t f_{soft}(x_t, x_{t-1}, \dots, a_t, a_{t-1}, \dots) \ \wedge \\
& \forall T \in [0, \Omega], \forall f_{strict}: \\
& \quad f_{strict}(x_T, x_{T-1}, \dots, a'_T, a'_{T-1}, \dots) \leq 0
\end{aligned}$$

In the following sections, we present and discuss solutions that aim at identifying such an optimal action sequence.

As mentioned, decision quality can suffer from inaccurate predictions. However, since predictions tend to increase decision quality, a_T should be chosen so that the decision is optimal regarding the current prediction.

Definition 4 (Optimal action regarding prediction)**.**
The action a_T is optimal regarding a prediction of future records iff the average of all soft requirement functions of the current and predicted points of time has the lowest value

possible under the condition that the current and all future actions fulfill all strict requirements.

4 Baselines

In this section, we present two baselines that are generalizations of related work to make decisions on streaming data w.r.t. requirements. We demonstrate that they do not meet our objectives and use them as competitors in our evaluation.

4.1 Single Predictions

To identify an optimal action a_T every time a new value x_T is recorded, one must consider future records. Our first baseline builds on related work [13], [14] that applies a prediction technique to gain information on the subsequent n future records. Having a series of predicted records $x_{T+1}, \dots, x_{T+n}$ at hand, the question is how to identify an optimal a_T.

We must formulate an optimization problem that excludes solutions that violate strict requirements and identifies the best solution among the remaining ones. We do not make any assumption regarding the form of the requirement functions, e.g., being convex or linear. This calls for a solution scheme for constrained problems that is sufficiently general. *Penalty methods* [29] do have this characteristic. They apply terms that describe the degree of fulfillment of each constraint, to transform a constrained optimization problem into an unconstrained one. They distinguish between two kinds of terms, namely *objective* and *penalty terms*:

The objective term quantifies the quality of feasible solutions.

Definition 5 (Objective term)**.**
Let $\min f(\circ)$ *be the objective of an optimization problem.*
o is an objective term iff

$$o(\circ) = f(\circ).$$

In our scenario, the objective term is the aggregation

$$\sum_{f_{\text{soft}}} \sum_{t} f_{\text{soft}}(\circ)$$

over all soft requirements. Thus, the objective term preserves the identity of soft requirements, i.e., the objective term and soft requirements define the same function. In contrast, the penalty term adds penalties if strict requirements are not fulfilled to transform a constrained optimization problem into an unconstrained one.

Definition 6 (Penalty term)**.**
Let $f_{strict}(\circ) \le 0$ be a constraint that restricts the solution space of an optimization problem. The function p is a penalty term iff

$$p(\circ) = \begin{cases} 0 & \textit{if } f_{strict}(\circ) \le 0 \\ \textsc{huge} & \textit{if } f_{strict}(\circ) > 0. \end{cases}$$

If the constraint $f_{\text{strict}}(\circ) \le 0$ is fulfilled, it does not have any influence on the objective function. Otherwise, it dominates the objective term so that the result of the objective function is larger than any feasible one, i.e., $\textsc{huge} \gg o(\circ)$. Normally, there is no information on the maximum of the objective term over all feasible solutions. Hence, an annealing penalty function [29] such as

$$\textsc{huge} = e^{f_{\text{strict}}(\circ)/\text{temp}}$$

is common, where temp $\in (0, 1]$ and, in case an identified solution is infeasible, the value of temp decreases until a feasible action is identified.

Penalty methods formulate an unconstrained optimization problem by summing up the objective and the penalty term.

Definition 7 (Penalty method)**.**
Let $o(\circ)$ be an objective term and $p(\circ)$ a penalty term. The minimum

$$\min o(\circ) + p(\circ)$$

is the objective of a penalty method.

We now instantiate the objective and penalty terms of penalty methods with requirement functions.

Baseline 1. *Let a set of strict and soft requirements with their corresponding functions f_{strict} and f_{soft} and a sequence of predicted records $x_{T+1}, \dots, x_{T+n}$ be given. At each point of time T, we solve the following optimization problem.*

$$\min_{a_T, \dots a_{T+n}} \sum_{t=T}^{T+n} \sum_{f_{strict}} p(f_{strict}(x_t, x_{t-1}, \dots, a_t, a_{t-1}, \dots)) + \sum_{f_{soft}} f_{soft}(x_t, x_{t-1}, \dots, a_t, a_{t-1}, \dots)$$

The decision we make at T is the action a_T.

Now we prove that the solution of the optimization problem is the optimal action according to Definition 4.

Lemma 1. *The solution a_T of Baseline 1 is optimal according to Definition 4.*

Proof. According to Definition 4, an optimal action has the following two characteristics: First, the current action a_T must not violate any strict requirement f_{strict}. Second, under the restrictions defined by f_{strict}, a_T fulfills the average over all soft requirements f_{soft} as well as possible. We now prove that our optimization problem fulfills both characteristics.

1) Suppose that our optimization problem identified a sequence of actions that violated at least one strict requirement. According to Definition 6, the result of the penalty function would be larger than the largest value of the objective term over all feasible solutions. In consequence, the identified solution is not minimal, and our optimization problem cannot identify infeasible actions.
2) Assume that there is a solution $a'_T, \dots, a'_{T+n}$ that fulfills all strict requirements and has a smaller aggregated value over all soft requirements compared to the identified solution $a_T, \dots, a_{T+n}$. Since our optimization problem considers strict and soft requirements within one function simultaneously, the solution $a'_T, \dots, a'_{T+n}$ would result in a smaller value than the solution $a_T, \dots, a_{T+n}$. In consequence, $a_T, \dots, a_{T+n}$ cannot be a solution of our optimization problem. □

Drawbacks. According to Lemma 1, the solution identified is optimal for the given prediction. If the prediction is inaccurate, the quality of the decision decreases as well. As illustrated in Example 1, the choice of the prediction technique depends on the requirements. The independence from any assumptions regarding the relationship between prediction techniques and requirements is a distinctive feature of our approach. I.e., we do not expect individuals who use our technique to have any knowledge on this relationship. Different techniques might provide additional information, cf. Example 2. Thus, we now look at Markov Decision Processes that consider several possible futures simultaneously.

4.2 Markov Decision Processes

In contrast to Section 4.1, we now assume that every time an individual records a new value x_T, we know a probability distribution over future records, provided by a Markov chain. A Markov chain is a stochastical process that consists of a finite set of states S_{chain} and probabilities $P_{\text{chain}}(s, s')$ to transit from state s at time T to state s' at time $T+1$. In our scenario, states in S_{chain} bear information on previous records, at least on x_T, and P_{chain} the probabilities to observe record x_{T+1} after x_T has been observed.

Example 7. *Consider a data stream with two possible values 0 and 1. Figure 3a features an example. After observing value* 0 *at* T*, there is a chance of* 80% *to observe the value* 0 *and one of* 20% *to observe* 1 *at* $T+1$*. To increase the prediction accuracy, one can include further information in the state description. For instance, in addition to the current value, this can be the current time of day ("observing* 0 *at* 9 *p.m.") or previous records ("observing* 0 *at* T *after observing 1 at* $T-1$*").*

We now extend the description of the temporal behavior provided by Markov chains to Markov Decision Processes (MDP) [30]. MDPs choose the best action at each state considering the distribution of expected future states.

Definition 8 (Markov Decision Process)**.**
The tuple

$$\langle S, A, P(s, a, s'), R(s, a, s') \rangle$$

is a Markov Decision Process where

- *S is a finite set of states,*
- *A is a finite set of actions,*
- *$P(s, a, s')$ are probabilities to transit from state $s \in S$ at time T to state $s' \in S$ at time $T+1$ when choosing action $a \in A$, and*
- *$R(s, a, s')$ are rewards for transitting from $s \in S$ to $s' \in S$ after choosing action $a \in A$*

Having defined MDP, the problem is to identify a *policy* π that specifies an action for each state that maximizes the cumulative reward. Approaches such as value iteration or reinforcement learning identify such an optimal policy [31].

We now illustrate the application of MDP to perturbation:

Example 8. *Think of the binary data stream in Example 7. In addition, consider the strict requirement "published records must have a binary value" and the soft requirement "minimize the quadratic distance between subsequent records".*

First, we describe the *state* and *action space* of the MDP:

Example 8.1. *To validate whether the strict requirement is fulfilled and to compute the quality of the soft requirement, the state description of the MDP includes the value of the previously published record y_{T-1} and of the current record x_T. In consequence, the MDP has four states $s_1 = \langle y_{T-1} = 0; x_T = 0 \rangle$, $s_2 = \langle y_{T-1} = 0; x_T = 1 \rangle$, $s_3 = \langle y_{T-1} = 1; x_T = 0 \rangle$ and $s_4 = \langle y_{T-1} = 1; x_T = 1 \rangle$. The actions at each state are deviations a_T where $a_T \in \{-1, 0, 1\}$. Figure 3b graphs the general structure of the MDP.*

We now formulate transition probabilites and rewards.

Example 8.2. *Figure 3c graphs the transition and Figure 3d the reward matrices of each action. The transition probabilities result from the initial Markov chain in Example 7. For instance, the probability to transit from state s_2 to s_1 choosing action -1 is $P(s_2, -1, s_1) = 0.6$. This is because the probability to transit from record $x_T = 1$ to record $y_{T+1} = 1 - 1 = 0$ is $P_{chain}(1, 0) = 0.6$. If an action resulted in an infeasible record such as $y_{T+1} = -1$ or $y_{T+1} = +2$, the transitions apply the probabilities to the closest feasible records, e.g., transiting to the published record* 0 *instead of* -1*. However, rewards reflect the infeasibility of an action. Here, infeasible actions result in the lowest possible reward, i.e., $R(s_1, -1, s_1) = -\infty$. Rewards of feasible actions result from soft requirements. For instance, the reward of reducing the current record* 1 *by* -1 *after publishing $y_{T-1} = 1$ at the previous point of time is $R(s_4, 1, s_1) = R(s_4, 1, s_2) = -(0-1)^2 = -1$.*

Approaches such as policy iteration [31] identify the optimal *policy* that returns the optimal action at each state. For the MDP in Example 8, the following policy is optimal:

$$\pi = \begin{array}{c} \begin{array}{cccc} 0;0 & 0;1 & 1;0 & 1;1 \end{array} \\ \begin{pmatrix} \pm 0 & -1 & +1 & \pm 0 \end{pmatrix} \end{array}$$

Now we explain how to model requirements using MDP.

Baseline 2. *Let a set of strict and soft requirements with their corresponding functions f_{strict} and f_{soft} be given. We identify the policy π of the MDP with the following structure:*

- *Set of states S: The state description must include the union of the inputs of all requirement functions f_{strict} and f_{soft}, excluding the current action a_T.*
- *Set of actions A: Actions cover the range of feasible actions a_T.*
- *Transition probabilities $P(s, a_T, s')$: The transition probabilities result from a Markov chain whose state description contains information identical to the one of the state description of the MDP, except for information on previous actions. The transition probability $P(s, a_T, s')$ is equal to $P_{chain}(s \setminus \{a_{T-1}\}, s' \setminus \{a_T\})$.*
- *Rewards $R(s, a_T, s')$: If any strict requirement is infeasible, the reward has the lowest possible value, here $-\infty$. Otherwise, the reward is $R(s, a_T, s') = -\sum_{f_{soft}} f_{soft}(s, a_T)$.*

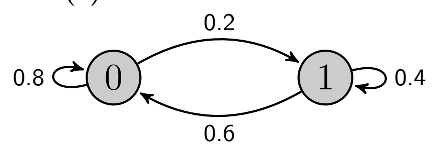
Let s_T be the current state of the MDP at T. The action $a_T = \pi(s_T)$ is the decision we make at T.

We now prove that the resulting optimal policy yields the decision of the highest quality according to Definition 4.

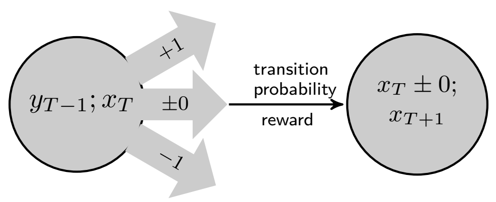
Lemma 2. *The solution a_T of Baseline 2 is optimal according to Definition 4.*

Proof. Since the optimal policy of a MDP results in the maximum reward, we now show that the actions of the policy result in an optimal decision quality according to Definition 4. I.e., we prove that the policy (1) does not violate any strict requirement, and that (2) it minimizes the average over all soft requirements as well as possible.

(a) Initial Markov Chain



(b) MDP Structure



(c) MDP Transition Matrices

Action −1

$$\begin{array}{c|cccc} & 0;0 & 0;1 & 1;0 & 1;1 \\ \hline 0;0 & 0.8 & 0.2 & 0.0 & 0.0 \\ 0;1 & 0.6 & 0.4 & 0.0 & 0.0 \\ 1;0 & 0.8 & 0.2 & 0.0 & 0.0 \\ 1;1 & 0.6 & 0.4 & 0.0 & 0.0 \end{array}$$

Action ±0

$$\begin{array}{cccc} 0;0 & 0;1 & 1;0 & 1;1 \\ \hline 0.8 & 0.2 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.6 & 0.4 \\ 0.8 & 0.2 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.6 & 0.4 \end{array}$$

Action +1

$$\begin{array}{cccc} 0;0 & 0;1 & 1;0 & 1;1 \\ \hline 0.0 & 0.0 & 0.8 & 0.2 \\ 0.0 & 0.0 & 0.8 & 0.2 \\ 0.0 & 0.0 & 0.6 & 0.4 \\ 0.0 & 0.0 & 0.6 & 0.4 \end{array}$$

(d) MDP Reward Matrices

Action −1

$$\begin{array}{c|cccc} & 0;0 & 0;1 & 1;0 & 1;1 \\ \hline 0;0 & -\infty & -\infty & - & - \\ 0;1 & 0 & 0 & - & - \\ 1;0 & -\infty & -\infty & - & - \\ 1;1 & -1 & -1 & - & - \end{array}$$

Action ±0

$$\begin{array}{cccc} 0;0 & 0;1 & 1;0 & 1;1 \\ \hline 0 & 0 & - & - \\ - & - & -1 & -1 \\ -1 & -1 & - & - \\ - & - & 0 & 0 \end{array}$$

Action +1

$$\begin{array}{cccc} 0;0 & 0;1 & 1;0 & 1;1 \\ \hline - & - & -1 & -1 \\ - & - & -\infty & -\infty \\ - & - & 0 & 0 \\ - & - & -\infty & -\infty \end{array}$$

Fig. 3. MDP example

1) We show that no solution that violates any strict requirement can be part of the policy. The proof is in line with the proof of Lemma 1: Actions that violate any strict requirement result in the lowest solution quality, in this scenario in the minimal reward $-\infty$. In consequence, the solution quality of each action that does not violate any strict requirement is higher, independently from the soft requirements. Thus, actions that violate any strict requirement cannot be part of the optimal policy.
2) The optimal policy minimizes the average over all f_{soft} as well as possible. This is because rewards determined from feasible actions result from the average over all soft requirement functions at the current point of time. The optimal policy specifies the actions that maximize the cumulative rewards. I.e., the reward is equal to the highest decision quality, as required in Definition 4. □

Drawbacks. We now explain that this structure of a MDP has several characteristics which curb decision quality.

1) MDPs require a finite state and action space. This makes a discretization of the data streams necessary and in consequence yields quality losses. In addition, the set of actions must be finite. This is in the way of applying MDPs to use cases such as differential privacy where deviations must follow an unrestricted distribution [12].
2) Transitions follow the Markov assumption. I.e., they depend on the actual state only and cannot include information on states visited previously. For high prediction accuracy, one can extend the state description. However, this requires precise knowledge of which information to add and leads to an explosion of the state space. Thus, identifying the optimal policy requires a huge body of training data and long runtimes.

We now propose a new approach in Section 5. It does not have the drawbacks of both baselines identified so far.

5 Decision Making using Multiple Predictions

We now propose a solution with the following features: (1) It is independent from any assumptions regarding the relationship between prediction techniques and requirements. (2) It considers multiple possible futures, without the need for discretization or complex instantiations of predictors.

5.1 Overview

Besides strict and soft requirements, our framework has a set M of prediction techniques as input. Since our framework identifies good techniques automatically, M may also contain techniques which often do not perform well, such as taking the value 24 hours or one week ago as prediction. Even though such techniques may result in large prediction errors, they tend to cover the value range. This can be helpful for some requirements, cf. Example 1.

Our framework works with optimization methods that solve unconstrained problems. In our evaluation, we have resorted to the Nelder-Mead method [32] that performs best for our data and requirements. But one is free to use any other approach such as hill climbing or genetic algorithms.

In addition, our framework requires two points of time t_1 and t_2 as input. They define three steps: (I) Over a period of time $[1, t_1]$, the framework trains each predictor. (II) In $[t_1+1, t_2]$, the framework computes an individual weight for each predictor. This weight is in line with the expected quality improvement of the predictor when being combined with others. (III) Starting at t_2+1, we use the weighted predictors to make decisions. To this end, we formulate a new optimization problem that considers all predictors simultaneously. We now present details of each step.

5.2 Step I: Training

Comparably to our baselines that require some training, our framework requires a data stream sample for the initial training. Its length is t_1. As a result of the first step, each technique m is capable of returning predicted records $x^m_{T+1}, x^m_{T+2}, \dots, x^m_{T+n}$ each time a new value x_T is recorded.

For now, we assume that the nature of our data stream remains constant. I.e., the predictors and their weights do not change over time. However, this assumption does not hold for cold starts or concept drifts [16]. Thus, Section 5.5 explains how to adapt our framework to cope with these challenges.

5.3 Step II: Weighting

In the second step, our framework determines a weight for each prediction technique according to its expected improvement when becoming part of an ensemble solution in the third step. We first explain the general weighting scheme and discuss its implementation afterwards.

Algorithm 1 Weight Determination

```
Input:  original records $x_t$ and predicted records $x_t^m$
          for $t \in [t_1+1, t_2]$ and $m \in [1, M]$,
        requirement functions $f_{\text{strict}}$ and $f_{\text{soft}}$
Output: weights $w_m$ for $m \in [1, M]$
 1: determine actions $a_t^m$ for predictions $x_t^m$ regarding Lemma 1
 2: determine actions $a_t^{\text{opt}}$ for records $x_t$ regarding Lemma 1
 3: for $t \in [t_1+1, t_2]$ do
 4:     $\text{qual}(\text{opt}, t) = -\sum_{f_{\text{soft}}} f_{\text{soft}}(x_t, x_{t-1}, \dots, a_t^{\text{opt}}, a_{t-1}^{\text{opt}}, \dots)$
 5:     for $m \in [1, M]$ do
 6:         $\text{qual}(m, t) = -\sum_{f_{\text{soft}}} f_{\text{soft}}(x_t, x_{t-1}, \dots, a_t^m, a_{t-1}^m, \dots)$
 7:         $\text{diff}(m, t) = (\text{qual}(\text{opt}, t) - \text{qual}(m, t))^2$
 8: normalize diff such that $\forall m, t : \text{diff}(m, t) \in [0, 1]$
 9: for $t \in [t_1+1, t_2]$ do
10:     $v = \text{median}(\{\text{diff}(m, t) | \forall m \in [1, M]\})$
11:     for $m \in [1, M]$ do
12:         $\text{diff}(m, t) = \text{diff}(m, t) - v$
13: initialize $z_t = 1/(t_2 - t_1)$ for $t \in [t_1+1, t_2]$
14: for $i \in [1, M]$ do
15:     choose $m$ that minimizes $\sum_t z_t \cdot \text{diff}(m, t)$
16:     identify $w_m$ that minimizes $\sum_t z_t \cdot e^{w_m \cdot \text{diff}(m,t)}$
17:     for $t \in [t_1+1, t_2]$ do
18:         $z_t = z_t \cdot e^{w_m \cdot \text{diff}(m,t)}$
19:     normalize $z$ such that $\sum_t z_t = 1$
20: return $w$
```

5.3.1 Weighting scheme

An obvious idea how to determine the weights is to consider the quality of prediction techniques. However, as Example 2 has shown, this has issues. As discussed, connections between prediction techniques and requirements are hard to detect and describe. It is even unreasonable to assume domain experts being able to make such connections explicitly. Thus, we propose a weight determination technique that includes requirements automatically.

We define the decision quality at a certain point of time t:

Definition 9 (Decision quality at t). *Let a set of strict and soft requirements with their corresponding functions f_{strict} and f_{soft} and sequences of records $x_0, \dots, x_t$ and actions $a_0, \dots, a_t$ be given. The decision quality at point of time t is*

$$qual(t) = \begin{cases} -\infty \\ \quad \text{if } \exists f_{strict} : f_{strict}(x_t, x_{t-1}, \dots, a_t, a_{t-1}, \dots) > 0 \\ -\sum_{f_{soft}} f_{soft}(x_t, x_{t-1}, \dots, a_t, a_{t-1}, \dots) \quad \text{else.} \end{cases}$$

We consider the decision quality of each technique at each point of time in $[t_1+1, t_2]$. By considering each point of time individually, different prediction techniques perform best at different points of time. In order to avoid overfitting, we do not determine the weights on the sequence of the data stream from Step I. To determine the decision qualities for each prediction technique, we solve the optimization problem from Lemma 1. In Step III, we formulate a new optimization problem that extends the one from Lemma 1 by combining several predictions. The extension combines the prediction techniques linearly. Our weight determination is an adaptation of established boosting mechanisms [17] that combine classifiers linearly. So we can use the decision quality of individual prediction techniques here to determine the weights.

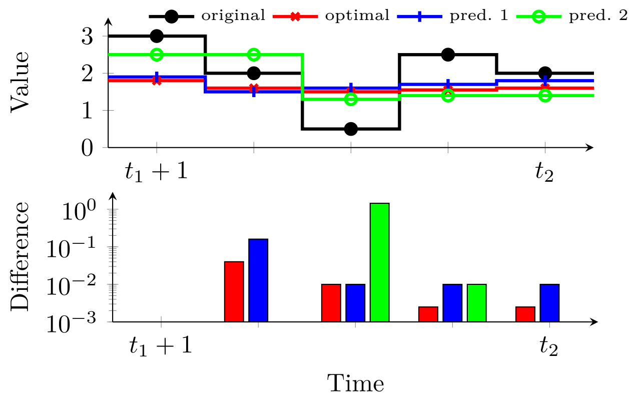


Fig. 4. Decision qualities for Example 9

5.3.2 Weighting algorithm

Algorithm 1 is our weight determination approach. We now explain its main parts.

Determine decision qualities (Line 1-6). We first solve the optimization problem from Lemma 1 for each prediction technique m (Line 1). In addition, we solve the optimization problem with original records as predictions (Line 2) in order to have an oracle. Now we compute the decision qualities at each point of time t for each prediction technique m (Line 4) and the original records (Line 6) according to Definition 9.

Comparing decision qualities (Line 7). Next, we compute the differences between the qualities of the prediction techniques and the oracle. Example 9 illustrates why we need the comparison with the oracle.

Example 9. *Think of some strict requirements and the soft requirement "minimize the quadratic difference between subsequent records", as in Example 5. The upper part of Figure 4 shows an original data stream (black line) as well as a perturbation that is optimal regarding the requirements (red line). In addition, there are two perturbations (blue and green lines) that base on two prediction techniques. The lower part compares the decision qualities of the three perturbations with each other. At t_1+1, the difference is undefined for all perturbations. This is because there is no previous perturbation the quadratic difference can be computed with. The cumulative difference over all points of time is lowest for the optimal perturbation (0.055). However, at t_1+2 and t_2, the perturbation based on Technique 2 is better. This local improvement results in a larger cumulative difference of Technique 2 (1.45) compared to Technique 1 (0.19). At t_1+3 in particular, the cumulative difference would be better if $a_{t_1+2}^{pred.\ 2}$ had a lower value.*

As Example 9 illustrates, choosing prediction techniques that result in best decisions at certain points of time can be misleading. Instead, by choosing prediction techniques that lead to actions most similar to actions that would result from perfect knowledge of the future, we put qualities at single points of time into the global context. We compute the quadratic distance $\text{diff}(t, m)$ between the decision qualities at each point of time t of each prediction technique m with the decision quality of the original records (Line 7).

Normalization (Line 8-12). We now apply a boosting mechanism [17] to determine a weight w_m for each prediction technique m. As proposed in [33], we use an exponential function to quantify the quality loss induced by the imperfectness of any prediction technique. First, we normalize the quality differences to be applicable to the error function: To identify a weight w_m that minimizes $\sum_t e^{w_m \cdot \text{diff}(t,m)}$, values of diff$(t,m)$ must be in $[-1,1]$. In consequence, we normalize the differences to range in $[0,1]$ (Line 8) and shift the medians of each point of time t to value 0 (Lines 9-12). This approach is robust in the presence of differences of extreme value.

Iterative Boosting (Line 13-19). Following the boosting idea of [33], we apply an iterative approach that considers prediction techniques successively. For every prediction technique, our approach compares the quality differences to optimal decisions at each point of time t with the differences of prediction techniques of earlier iterations. The weight of the current prediction technique depends on its ability to improve the quality at those points of time previous techniques resulted in large differences. In the beginning, there is no information on differences. Thus, we create weights z_t for each t that have the same initial value (Line 13). We iteratively identify the prediction technique m with the smallest qualitfy difference to the optimal actions $\sum_t z_t \cdot \text{diff}(m,t)$ (Line 15). For this technique, we compute its weight w_m that minimizes the term $\sum_t z_t \cdot e^{w_m \cdot \text{diff}(m,t)}$. Finally, we update the temporal weights z_t in Lines 17-19 to have a higher weight in the next iteration if the current prediction technique m has a large decision-quality difference to the perfect decision at t and a smaller weight vice versa. After m iterations, we have identified a weight w_m for each prediction technique m. As illustrated in [17], this allows to weight techniques that perform best in different cases to create an ensemble solution.

5.4 Step III: Making Decisions

In Step III, we apply the pre-trained prediction techniques, taking their weights into account, to make decisions. To do so, we formulate a new optimization problem that relies on penalty methods. The problem is similar to Baseline 1, but it takes several series of predictions into account.

Framework. *Let a set of strict and soft requirements with their corresponding functions f_{strict} and f_{soft}, a set of prediction techniques M with corresponding weights w_m ($m \in [1, M]$) and sequences of predicted records $x^1_{T+1}, \dots, x^1_{T+n}, \dots, x^M_{T+1}, \dots, x^M_{T+n}$ be given. At each point of time T we solve the following optimization problem:*

$$qual(t) = \begin{cases} -\infty \\ \quad if\, \exists f_{strict} : f_{strict}(x_t, x_{t-1}, \dots, a_t, a_{t-1}, \dots) > 0 \\ -\sum_{f_{soft}} f_{soft}(x_t, x_{t-1}, \dots, a_t, a_{t-1}, \dots) \quad else. \end{cases}$$

The action a_T that froms the solution is the decision at T.

Now we prove that the solution of the optimization problem is the optimal action according to Definition 4.

Lemma 3. *The solution a_T of our framework is optimal according to Definition 4.*

Proof. For $M = 1$, the proof is identical to the proof of Lemma 1. For $M > 1$, the proof results from the proof of Lemma 1, i.e., the identified solution $a_T, a^1_{T+1}, \dots, a^M_{T+n}$ (1) does not violate any strict requirement for any series $x_T, x^m_{T+1}, \dots, x^m_{T+n}$ for $m \in [1, M]$ and (2) fulfills the average over all soft requirements over all predictions according to their weights w_m as well as possible. □

Since we do not create an aggregated series over all predictions, but solve the problem for all predictions simultaneously, there is an action variable $a^m_{t'}$ for each prediction technique $m \in [1, M]$ at each point of time $t' \in [T+1, \dots, T+n]$. However, all predictions share the same currently observed record x_T and its action variable a_T. Thus, we optimize a_T at the current point of time T w.r.t. all predictions and their weights w_m. This means that we meet the required properties: (1) in contrast to using a single prediction, we can apply arbitrary prediction techniques and do not depend on knowing the connection between prediction techniques and requirements. (2) in contrast to MDPs, we solve an optimization problem to identify the best action that does not require any discretization or results in an exploding state space.

5.5 Further Refinements: Updates over Time

Until here, we assumed that training the prediction techniques at the beginning is feasible, and that the behavior of data streams does not change over time. However, this does not hold if a cold start is necessary, or if concepts shift. These challenges are there in any case, i.e., for our baselines as well as for our framework. We now explain how to adapt our baselines and our framework to cope with cold starts and concept shifts.

If a cold start is necessary, we initially use random predictions and weights. To improve quality over time, we store all records in the starting phase. At each point of time, we perform the training and weighting using all stored records. When the starting phase is over, our approaches will work as described before.

If the characteristics of the data stream are non-stationary, we need to update predictors and their weights over time. For MDPs, there exist solutions that are directly applicable to our problem [34]. When it comes to updating the predictors, some approaches propose to partition the data in sequential blocks [18] or data chunks [19]. These partitions can either be used to update predictors [20] or to train and replace existing ones [18]. Analogously to updating the predictors, we can also re-compute the weights every time we update the predictors. An alternative to regular updates are techniques that detect concept drifts and perform updates only when needed [22].

6 Experiments

We evaluate the decision quality of the two baselines and of our framework. Prediction techniques applied for the first baseline as well as for our framework are as follows: Simple Exponential Smoothing (SES), Double Exponential Smoothing (DES), Double Seasonality Exponential Smoothing (DSES), Pattern Sequence based Forecasting (PSF), Pattern Sequence based Forecasting with temporal information (PSFtemp), artificial Neural Network (aNN), Floating Average (FA), "same value as previous day" (prevDay) and "same value as previous week" (prevWeek). In addition, we generate an ensemble prediction (Aggregation) [10] by aggregating over all predictions using the weights identified by Algorithm 1 for Baseline 1.

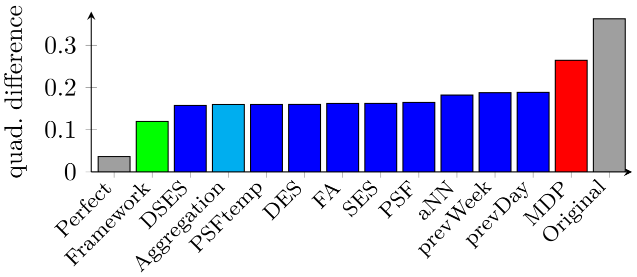


Fig. 5. Distances of Use Case 1

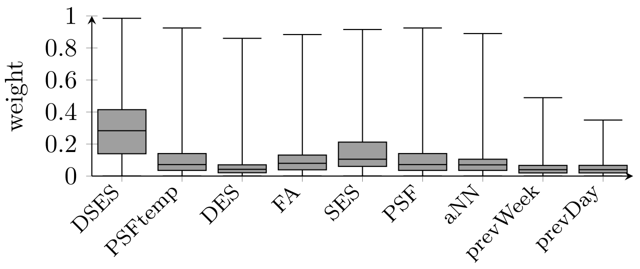


Fig. 6. Weights of Use Case 1

To solve the optimization problems of Lemmas 1 and 3, we apply the Nelder-Mead method [32] with parameters $\alpha = 1.0$, $\gamma = 2.0$, $\beta = 0.5$ and $\sigma = 0.25$. To identify the policy of our MDPs, we apply the policy-iteration approach [31] with a discount factor of 0.9.

6.1 Use Case 1: Perturbation via Batteries

Experiment Setup. For our first use case, we deploy batteries whose charging perturbs the energy consumed [7]. As illustrated in Example 4, the maximum charging rate and the capacity restrict the perturbation. In addition, we add the soft requirement to minimize the quadratic difference of subsequent records to hide activities, as explained in Example 5.

The optimization problems result from the requirement functions presented in Examples 4 and 5. The state description of the MDP contains a discretized representation of the current record x_T, a discretized representation of the published record y_{T-1} of the previous point of time and a discretized representation of the current load level $c_T = \sum_{t=0}^{T} a_t$. The action space contains discretized deviations whose absolute value $|a_t|$ is smaller than the max. charging rate $p_{\max}$. On average, each MDP has 12.8 transition and 12.8 reward matrices, each consisting of about 26 million cells. Due to this huge complexity, we update the probability transitions and optimal policies only once a week (every 336 points of time).

We perform our experiments for the first use case on the CER dataset [35]. This is because it contains private energy consumptions of over 2500 households recorded every 30 minutes for almost 1.5 years. Due to the large number of records per data stream, we are able to train our prediction techniques precisely, and due to the large number of data streams, we expect our results to be robust.

Results. Table 1 shows the average runtimes of each approach and use case to perturb a data stream, Table 2 shows runtime statistics about the perturbation of single records. MDPs require on average 129 minutes per household. I.e., every time we update the Markov chain and compute the optimal policy, it takes over 100 seconds. Accessing the computed policy in turn requires almost no time. Solving the optimization problem of Lemma 1 requires about 2 min for an entire data stream. In contrast, the ensemble from Lemma 3 takes about 30 min. This is because the number of variables the optimization problem considers is linear with the number of prediction techniques. However, with less than 0.1 seconds to compute a deviation a_T, our framework is well applicable in a scenario where one value is recorded every 15 min.

Figure 5 shows the average quadratic differences of subsequent records of all approaches. The first bar is the average difference one can achieve with perfect knowledge on the future. The last bar shows the differences in the original unperturbed data. As a first result, we observe that all approaches reduce the differences of the original record. MDPs perform worse than perturbations based on single predictions and our framework. We see the reason for this in the discretization of the records and actions, as explained in Section 4.2. The quadratic differences of the perturbations that rely on single predictions perform similarly and have a maximum difference of 0.03 to each other. As expected, creating an averaged solution over all predictions does not increase decision quality, and DSES provides slightly better results. Our framework clearly outperforms its competitors.

To analyze the influence different prediction techniques have on our framework, we summarize their weights in Figure 6. Their order is identical to their order in Figure 5. We observe that the weights are not monotonically decreasing

	Baseline 1	Baseline 2	Framework
Use Case 1	128	7,789	1,817
Use Case 2	58	NA	413
Use Case 3	27,632	NA	436,584

TABLE 1
Average Runtimes per Use Case [sec]

		Baseline 1	Baseline 2	Framework
Use Case 1	min	0.002	0.001	0.004
	avg	0.009	0.383	0.092
	max	0.114	128.583	1.616
	std	0.009	1.64	0.070
Use Case 2	min	0.001	NA	0.003
	avg	0.008	NA	0.056
	max	0.097	NA	1.253
	std	0.005	NA	0.041
Use Case 3	min	0.081	NA	0.294
	avg	3.095	NA	48.901
	max	70.563	NA	788.185
	std	4.267	NA	92.371

TABLE 2
Runtimes per individual record [sec]

	Aggregation	SES	DES	DSES	PSF	PSFtemp	aNN	FA	prevDay	prevWeek	MDP
UC 1	31.97%	33.63%	32.34%	30.88%	34.76%	32.12%	42.58%	33.48%	45.03%	44.60%	63.28%
UC 2	23.06%	19.78%	37.12%	16.21%	13.39%	16.23%	27.58%	21.85%	36.29%	31.26%	NA
UC 3	5.49%	24.76%	29.85%	3.67%	20.17%	13.27%	18.29%	26.00%	59.90%	47.74%	NA

TABLE 3
Quality improvement of our framework compared to baselines

with an increasing difference. As expected, our weighting scheme does not allocate weights depending on the cumulative perturbation quality over the complete data stream, but combines techniques that perform best in different cases.

6.2 Use Case 2: Differential Privacy

Experiment Setup. In our second use case, we evaluate our framework in the presence of differential-privacy guarantees [12]. The scenario is one where an attacker disaggregates the energy consumption to devices to learn daily routines of individuals. If the perturbation follows a certain distribution [4], such attacks are guaranteed to fail. We implement the strict constraint described in Example 3. In addition, we add the soft constraint that the quadratic difference to the record three time stamps from now should be as small as possible. In consequence, we achieve a long-term smoothing. For our experiments, we use the REDD [36] and the smart* [37] datasets that contain information on the energy consumptions of single devices.

Results. First, we see that we cannot apply MDPs due to computational restrictions. This is because the state descriptions become daunting in size. To perform the statistical test whether the deviations follow a certain distribution, each state includes a histogram of the current distribution with b bins, each of them having one of v possible discretized values. Think again of Use Case 1: Here, the state description consists of only three parameters, and the description results in matrices with over 4 billion cells overall. For the current use case, the matrices would increase cubically multiplied with the number of bins. All this explains why it has not been possible to perform experiments for MDPs for this second use case.

We use the following measure to quantify the quality improvement our framework provides, by comparing it to a method m

$$\mathrm{QI}(m) = \frac{\mathrm{qual}(m) - \mathrm{qual}(\mathrm{framework})}{\mathrm{qual}(m) - \mathrm{qual}(\mathrm{perfect})}$$

where qual(m) is the quality of method m, qual(framework) the quality of our framework and qual(perfect) the quality when having perfect knowledge on the future. If $\mathrm{QI}(m) = 0\%$, our framework does not provide any improvement over method m. The second row of Table 3 shows the improvement of our framework compared to the different predictions with our first baseline. Since all values are positive, our framework outperforms its competitors. In contrast to Use Case 1, PSF provides the best results among the single predictions. This result indicates that our hypothesis that different prediction techniques perform best for different requirements is correct.

6.3 Use Case 3: Data Utility

Experiment Setup. For our third use case, we consider health-care data that allows inferences on sporting activities. Here, it is important to give guarantees that sudden changes in the data stream remain hidden. Thus, we formulate the strict requirement that differences in values of subsequent records do not exceed a threshold. At the same time, we want data to be useful to detect accidents, i.e., we formulate the soft requirement that the quadratic error of the perturbation should be as small as possible, cf. Example 6. Here, we use fitness data recorded by a pedometer every 5 minutes over 31 days [38].

Results. Again, we cannot apply MDPs. The reason is that it is not possible to define an action space that guarantees the inclusion of any feasible deviation. For instance, if the data stream increases over a long period of time, feasible deviations might become larger and larger. Since we do not have any knowledge on lower or upper limits of future records, we cannot implement a MDP for this use case.

Considering the quality improvement in Table 3, we observe that our framework again outperforms the baselines. We observe that the prediction technique DSES clearly outperforms the other elementary techniques. The information gain our framework obtains from other techniques is small compared to the previous use cases. This result shows that choosing good prediction techniques is still an important aspect, but our framework can identify the best techniques to create a perturbation that is at least as good as the perturbations that result from single prediction techniques.

7 Conclusions

This paper has studied decision-making on data streams in the presence of requirements. We have proposed formulations of strict and soft requirements and have shown that our language is capable of covering related work, e.g., technical, utility or privacy requirements on data perturbation. Next, we have generalized approaches used in related work, namely solving an optimization problem based on a single prediction and solving MDPs. We see that both approaches have different drawbacks, such as requiring knowledge on the relation between requirements and prediction techniques or loosing decision quality from discretization. In consequence, we have proposed a new framework that does not suffer from these drawbacks. It can handle multiple predictions simultaneously. I.e., in scenarios where no accurate prediction is possible, our framework covers the range of possible future records. To consider the varying impacts of different prediction techniques on the decision quality, we have proposed an ensemble scheme that determines weights for each prediction. By solving an optimization problem, one can determine the best decision for the given predictions. Experiments on real-world data show the applicability to different use cases and the quality improvement of our solution. For future research, we plan to explore further requirements from different domains and to extend our solution to multidimensional data streams.

References

[1] E. P. Kasten and P. K. McKinley, "Meso: Supporting online decision making in autonomic computing systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 4, pp. 485–499, 2007.

[2] E. McKenna, I. Richardson, and M. Thomson, "Smart meter data: Balancing consumer privacy concerns with legitimate applications," *Energy Policy*, vol. 41, pp. 807–814, 2012.

[3] S. Papadimitriou, F. Li, G. Kollios, and P. S. Yu, "Time Series Compressibility and Privacy," in *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB)*, 2007, pp. 459–470.

[4] G. Ács and C. Castelluccia, "I Have a DREAM! (DiffeRentially privatE smArt Metering)," in *Information Hiding - 13th International Conference (IH)*, 2011, pp. 118–132.

[5] Y. Xiao, L. Xiong, S. Zhang, and Y. Cao, "LocLok: Location Cloaking with Differential Privacy via Hidden Markov Model," *Proceedings of the VLDB Endowment*, vol. 10, no. 12, pp. 1901–1904, 2017.

[6] S. E. McLaughlin, P. McDaniel, and W. Aiello, "Protecting consumer privacy from electric load monitoring," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, 2011, pp. 87–98.

[7] G. Kalogridis, C. Efthymiou, S. Z. Denic, T. A. Lewis, and R. Cepeda, "Privacy for smart meters: Towards undetectable appliance load signatures," in *2010 First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2010, pp. 232–237.

[8] F. Martínez-Álvarez, A. Troncoso, G. Asencio-Cortés, and J. C. Riquelme, "A Survey on Data Mining Techniques Applied to Electricity-Related Time Series Forecasting," *Energies*, vol. 8, no. 11, pp. 13 162–13 193, 2015.

[9] X. Liu and P. S. Nielsen, "Scalable Prediction-based Online Anomaly Detection for Smart Meter Data," *Information Systems*, vol. 77, pp. 34–47, 2018.

[10] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and systems magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[11] M. Przybyla-Kasperek and A. Wakulicz-Deja, "Dispersed decision-making system with fusion methods from the rank level and the measurement level – A comparative study," *Information Systems*, vol. 69, pp. 124–154, 2017.

[12] C. Dwork, "Differential Privacy," in *Automata, Languages and Programming, 33rd International Colloquium (ICALP)*, 2006, pp. 1–12.

[13] S. Percy, M. Aldeen, and A. Berry, "Residential precinct demand forecasting using optimised solar generation and battery storage," in *IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*, 2015, pp. 1–5.

[14] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Short-term traffic forecasting: Where we are and where we're going," *Transportation Research Part C: Emerging Technologies*, vol. 43, no. 1, pp. 3–19, 2014.

[15] R. Jursa and K. Rohrig, "Short-term wind power forecasting using evolutionary algorithms for the automated specification of artificial intelligence models," *International Journal of Forecasting*, vol. 24, no. 4, pp. 694–709, 2008.

[16] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, vol. 37, pp. 132–156, 2017.

[17] R. E. Schapire and Y. Freund, *Boosting: Foundations and algorithms*. MIT press, 2012.

[18] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001, pp. 377–382.

[19] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 226–235.

[20] G.-B. Huang, N.-Y. Liang, H.-J. Rong, P. Saratchandran, and N. Sundararajan, "On-line sequential extreme learning machine," *Computational Intelligence*, vol. 2005, pp. 232–237, 2005.

[21] N. C. Oza, "Online bagging and boosting," in *2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 3. Ieee, 2005, pp. 2340–2345.

[22] P. Li, X. Hu, Q. Liang, and Y. Gao, "Concept drifting detection on noisy streaming data in random ensemble decision trees," in *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. Springer, 2009, pp. 236–250.

[23] E. B. Iversen, J. M. Morales, and H. Madsen, "Optimal charging of an electric vehicle using a markov decision process," *Applied Energy*, vol. 123, pp. 1–12, 2014.

[24] L. G. Jaimes, M. Llofriu, and A. Raij, "A stress-free life: just-in-time interventions for stress via real-time forecasting and intervention adaptation," in *Proceedings of the 9th International Conference on Body Area Networks*, 2014, pp. 197–203.

[25] P. Rashidi, D. J. Cook, L. B. Holder, and M. Schmitter-Edgecombe, "Discovering activities to recognize and track in a smart environment," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 4, pp. 527–539, 2011.

[26] J. Zhao, T. Jung, Y. Wang, and X. Li, "Achieving differential privacy of data disclosure in the smart grid," in *INFOCOM, 2014 Proceedings IEEE*, 2014, pp. 504–512.

[27] S. R. Rajagopalan, L. Sankar, S. Mohajer, and H. V. Poor, "Smart meter privacy: A utility-privacy framework," in *IEEE Second International Conference on Smart Grid Communications (SmartGridComm)*, 2011, pp. 190–195.

[28] F. Laforet, E. Buchmann, and K. Böhm, "Individual privacy constraints on time-series data," *Information Systems*, vol. 54, pp. 74–91, 2015.

[29] A. A. Törn and A. Zilinskas, *Global Optimization*, ser. Lecture Notes in Computer Science. Springer, 1989, vol. 350.

[30] R. A. Howard, "Dynamic Programming and Markov Processes," *The M.I.T. Press*, vol. 132, no. 3428, pp. 667–667, 1960.

[31] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[32] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[33] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *European conference on computational learning theory*, 1995, pp. 119–139.

[34] M. Kwiatkowska, D. Parker, and H. Qu, "Incremental quantitative verification for markov decision processes," in *2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)*. IEEE, 2011, pp. 359–370.

[35] "Customer behaviour trials findings report (CER11/080a)," Commission for Energy Regulation (CER), Tech. Rep., 2011.

[36] J. Z. Kolter and M. J. Johnson, "Redd: A public data set for energy disaggregation research," in *Workshop on Data Mining Applications in Sustainability (SIGKDD)*, vol. 25, 2011.

[37] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, and J. Albrecht, "Smart*: An open data set and tools for enabling research in sustainable homes," 2012.

[38] D. Maurath, "RPubs - Analyzing FitBit Data," *https://rpubs.com/dmaurath/24643*, 2014.

Fabian Laforet is a research scientist and PhD student at Karlsruhe Institute of Technology (KIT), Germany. In 2012 he received the MSc in Information Engineering and Management at KIT. His current research interests include predictive analytics and data privacy with a focus on modern energy systems as well as privacy attacks.

Christian Olms works at the Fraunhofer Institute for Material Flow and Logistics. In 2018 he received his MSc in Computer Science at KIT and in 2013 his BSc in Applied Computer Science – Systems Engineering at the University of Duisburg-Essen, Germany. His research interests are data science and software engineering.

Rudolf Biczok is a Software Engineer working at Baloise Insurance in Basel. He graduated from KIT as a Computer Scientist after receiving his MSc in 2019 and his BSc in 2017. In his work, he focus on automation and digital transformation.

Klemens Böhm is full professor (chair of databases and information systems) at the KIT, since 2004. Prior to that, he has been professor of applied informatics/data and knowledge engineering at University of Magdeburg, Germany, senior research assistant at ETH Zürich, Switzerland, and research assistant at GMD – Forschungszentrum Informationstechnik GmbH, Darmstadt, Germany. Current research topics at his chair are knowledge discovery and data mining in big data, data privacy and workflow management.