Convex quadratic programming-based predictors: An algorithmic framework and a study of possibilities and computational challenges

Linnéa Gyllberg, Shudian Zhao[0000−0001−5949−9465], and Jan Kronqvist*[0000−0003−0299−5745]

Department of Mathematics, KTH Royal Institute of Technology, Stockholm, Sweden
* Corresponding author: Jan Kronqvist, jankr@kth.se

Abstract. We present a class of predictive models for forecasting time-series data, referred to as convex quadratic programming-based (CQPB) predictors. The predictions are computed from the minimizer of a convex quadratic problem, where previous observations are integrated as parameters. The remaining parameters, including constraints and objective coefficients, are trainable parameters. This work investigates the predictive capabilities of CQPB predictors and the computational challenges in their training. We analyze their properties and prove that this class of predictors includes classical autoregressive (AR) models, thus forming a generalization of AR models. The training problem is formulated as a bilevel optimization problem. To solve these training problems efficiently, we propose a two-stage heuristic algorithm based on the block coordinate descent approach. The results highlight the potential of CQPB predictors. Although training is challenging, our approach efficiently computes good solutions for moderate-size datasets.

Keywords: Time-series prediction · Inverse optimization · Bilevel optimization · Optimization-based predictive models.

1 Introduction

Prediction, or forecasting, is one of the fundamental challenges in data analysis, machine learning (ML), and artificial intelligence (AI). We focus on the prediction of time-series data, where the goal is to compute a prediction, or forecast, of the next point using a set of previous observations from the time series. However, the proposed framework is not limited to time-series data. Prediction of time-series data has been an active research area for more than fifty years [25]. A wide range of methods have been proposed over the years, ranging from classical methods such as autoregressive (AR) models [3] and autoregressive integrated moving average (ARIMA) models [7] to more modern approaches based on methods, such as support vector machines (SVM) [22], random forests (RF) [8], neural networks (NN) [12], and recurrent neural networks (RNN) [16].

In this paper, we present a class of predictors that are based on a convex quadratic program (QP). The prediction is given by the minimizer of a convex QP, and we refer to these as convex quadratic programming-based models. Some of the parameters in the QP are given by previous observations in the time series, and some are trainable parameters. We show that this forms a versatile class of predictive models and that it forms a generalization of AR models for certain parameter configurations. For certain applications, the time series can originate from a sequence of optimization problems, *e.g.*, prices in energy markets [5]. Even though the time series originates from the solutions of more complex optimization problems, it seems natural to compute the prediction from an optimization problem. Compared to, *e.g.*, neural network models, the QP-based prediction model can also offer a somewhat higher degree of interpretability.

Recently, there has been research on integrating convex optimization in neural networks [17, 1], *e.g.*, forming layers where the output is given by the minimizer of a convex problem. This line of work is related, but they focus on more general structures and different training procedures. Training the QP-based models can also be viewed as an inverse optimization problem [9], and there are similarities with studies (*e.g.*, [13, 14, 2, 10]) where they learn optimization problems.

The main contribution and goals of this paper can be summarized as:

- We propose a class of optimization-based predictors where the prediction is computed by a quadratic programming problem.
- We investigate the predicting capabilities of the proposed predictors and show that this class of predictors forms a generalization of AR models.
- We investigate the computational challenges of training convex optimization-based predictors through a so-called KKT reformulation. We propose some simple symmetry-breaking constraints to improve the computational performance and a simple heuristic decomposition technique to compute good solutions with a reasonable computational burden.

The goal is not to claim that the proposed class of predictors is superior but to investigate their potential. Training the convex optimization-based predictor through a KKT reformulation is obviously going to be challenging. However, due to the remarkable progress in state-of-the-art MIP solvers, it is not clear how much data or how large the predictive model can be for the training problem to remain solvable. To the authors' best knowledge, there is not much existing in the literature about predictors of this type nor about their predictive capabilities. One of the main goals has, therefore, been to analyze the predictive capabilities of this class of predictive models to determine if further research on solving the training problems more efficiently is motivated. The short answer is yes, and in the numerical experiment section, we show that the models have good predictive capabilities, but training them is challenging.

2 Convex Optimization-based predictors

Here we focus on the prediction of time-series data, where the goal is to estimate $\hat{\mathbf{x}}_{t+1} \in \mathbb{R}^d$ given a set of observations $\mathbf{x_t}, \mathbf{x_{t-1}}, \ldots, \mathbf{x_{t-k}} \in \mathbb{R}^d$. We propose

a somewhat unusual class of predictors where the prediction is computed from the minimizer of a convex optimization problem, whose parameters include the previous observations $\mathbf{x_t}, \mathbf{x_{t-1}}, \ldots, \mathbf{x_{t-k}}$. We are not the first to propose integrating convex optimization for prediction/estimation types of task, *e.g.*, see [2]. However, as it is a fairly recent methodology, we start by formally defining what we mean by a convex optimization-based prediction.

Definition 1. *A prediction* $\hat{\mathbf{x}}_{t+1}$ *computed by* $\hat{\mathbf{x}}_{t+1} = h(\mathbf{y}^*)$ *and*

$$\begin{aligned} \mathbf{y}^* = \underset{\mathbf{y}\geq\mathbf{0}}{\arg\min} \quad & f(\mathbf{y}) \\ s.t. \quad & g(\mathbf{y}, \mathbf{x_t}, \ldots, \mathbf{x_{t-k}}) = \mathbf{0}, \end{aligned} \tag{1}$$

where (1) *is a convex optimization problem, is regarded as a convex optimization-based prediction.*

Inference of the convex optimization-based predictor is computationally cheap if problem (1) is "well-behaved" and relatively small (convexity alone does not guarantee easy to solve). Compared to, *e.g.*, neural network models, inference can be more expensive as (1) does not have a closed-form solution. However, the main challenge is clearly in training the predictive model (1), *i.e.*, optimizing the functions f, g, and their coefficients to get an optimal prediction. Training is far from trivial as the training task results in a nonconvex optimization problem, more specifically, a bilevel optimization problem. For more details on bilevel optimization, we refer to [18, 4]. We refer to problem (1) as the inner optimization problem and the task of optimizing f, g as the training problem.

For simplicity and ease of presentation, we only consider single-variate time-series prediction, *i.e.*, $d = 1$, and $\hat{x}_{t+1} \in \mathbb{R}_+$. However, the techniques can also be applied to multi-variate prediction and can consider multiple input variables by simply restructuring the inner optimization problem (1) to incorporate these features. Also, note that the data can always be shifted to make all points positive. To keep the training problem tractable, we only consider *convex quadratic programming-based (CQPB)* predictors given by

$$\begin{aligned} CQPB(\{x_i\}_{i=t-k+1}^{t}; \mathbf{A}, \mathbf{b}, \mathbf{c}) := \hat{x}_{t+1} = & h(\mathbf{y}^*), \\ \mathbf{y}^* = \underset{\mathbf{y}\in\mathbb{R}^n}{\arg\min} \ & \mathbf{c}^\top \mathbf{y} + \frac{1}{2}\sum_{i=2}^{n} y_i^2 + \frac{1}{2}(y_1 - x_t)^2 \\ \text{s.t. } \mathbf{A}\mathbf{y} = & \begin{bmatrix} x_{t-k+1} \\ \vdots \\ x_t \\ \mathbf{b} \end{bmatrix}, \ \mathbf{y} \geq \mathbf{0}, \end{aligned} \tag{2}$$

where $\mathbf{c}$, $\mathbf{b}$, and $\mathbf{A}$ are trainable parameters with $k > 1$. Furthermore, we restrict the predictions to be given by

$$\hat{x}_{t+1} = h(\mathbf{y}^*) = [1 \ \ 0 \ldots \ \ 0]\mathbf{y}^*. \tag{3}$$

These restrictions are mainly to keep the resulting training problem computationally easier, but we will show that this is still a fairly general class of predictors capable of good predictive performance.

Before we continue, we want to briefly motivate this class of predictive models, *i.e.*, the structure of inner problem (2). The quadratic terms in the objective are included to ensure a unique optimizer of the problem, which avoids additional difficulties in training the parameters. The quadratic terms also act as a regularization and favors predictions close to the last observation.

Next, we briefly discuss the versatility of the predictive model given by (2). A simple but interesting result is that we can show that it forms a generalization of AR models [3]. We formalize this property in the following proposition.

Proposition 1. *The class of predictors given by* (2)*, where* $\mathbf{c}, \mathbf{b}$*, and* $\mathbf{A}$ *are model parameters, can be viewed as a generalization of autoregressive (AR) models.*

Proof. Consider the case with $\mathbf{b} = 0$, an arbitrary vector $\mathbf{c} \in \mathbb{R}^n$, and

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & -1 & \dots & -1 \\ 0 & \omega_{n-1} & 0 & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \dots & & & \omega_0 \end{bmatrix}. \tag{4}$$

The prediction given by (2) with this choice of $\mathbf{A}$ matrix, is simply $\hat{x}_{t+1} = \sum_{i=0}^{n-1} \omega_i x_{t-i}$. This shows that by choosing a certain structure of the inner problem, *i.e.*, the size of the $\mathbf{A}$ matrix, the predictor can be equivalent to an AR model with the same number of previous observations. But, when specifying the size of the $\mathbf{A}$ matrix, the training problem is not restricted to the specific $\mathbf{A}$ matrix given in (4), and can, therefore, be considered a generalization. □

Understanding the model complexity is important to prevent over-fitting. The number of parameters in (2) gives some insight into the model complexity, but to get a better understanding, we need to analyze the structure of problem (2). For example, if the RHS of the equation system in (2) contains as many old data points as the dimension of $\mathbf{y}$ and $\mathbf{A}$ has full rank, then the prediction is always given by a specific linear combination of old data points. We mainly consider the case where there is at least one more variable than the number of rows in $\mathbf{A}$, where the minimization operator plays a clear role in the prediction. With more variables in problem (2), we get a higher dimensional feasible set and more degrees of freedom. For a certain number of rows in $\mathbf{A}$ and a given number of old data points, more degrees of freedom in (2) generally result in a more complex predictive model.

3 Training CQPB predictors

Given the data $\mathbf{x} = \begin{bmatrix} x_1 \cdots x_N \end{bmatrix} \in \mathbb{R}_+^N$, the task of training the convex quadratic programming-based (CQPB) predictor, *i.e.*, determining the coefficients in $\mathbf{A}, \mathbf{b}$,

and $\mathbf{c}$, can be formulated as

$$\begin{aligned} \min_{(\mathbf{A},\mathbf{b},\mathbf{c})\in\Omega} \quad & \sum_{t=k}^{N-1} \left(x_{t+1} - \hat{x}_{t+1}\right)^2 \\ \text{s.t. } & \hat{x}_{t+1} = CQPB(x_{t-k+1},\ldots,x_t;\mathbf{A},\mathbf{b},\mathbf{c}),\ \forall t = k,..,N-1, \end{aligned} \tag{5}$$

where k is the number of observations used for each prediction. The set Ω is a simple box defined by upper and lower bounds of each entry of $\mathbf{A}, \mathbf{b}$ and $\mathbf{c}$. For simplicity and to keep the training easier, we set the bounds as ± 1. The training problem (5) is a so-called QP-QP bilevel problem. As the lower-level problem has a strictly convex quadratic objective function and only linear constraints, it satisfies the linear constraint qualification (LCQ) and has a unique minimizer. We can, therefore, use the standard approach of replacing the inner problems with their KKT conditions, *e.g.*, see [11, 14, 18]. The training problem can then be written as the single-level optimization problem

$$P_{train}(\mathbf{x},k,n,n_b) := \underset{\mathbf{A},\mathbf{c},\mathbf{y}_k,\ldots,\mathbf{y}_{N-1}}{\arg\min} \sum_{t=k}^{N-1} (x_{t+1} - \begin{bmatrix}1\,0\cdots\,0\end{bmatrix}\mathbf{y}_t)^2 \tag{6a}$$

$$\begin{aligned}
\text{s.t. } & -1 \le A_{j\ell} \le 1, && \forall j \in [k+n_b],\ \forall \ell \in [n], && \text{(6b)}\\
& -1 \le b_j \le 1, && \forall j \in [n_b], && \text{(6c)}\\
& -1 \le c_\ell \le 1, && \forall \ell \in [n], && \text{(6d)}\\
& \mathbf{A}\mathbf{y}_t = \bar{\mathbf{b}}_t, && \forall t \in \{k,k+1,\ldots,N-1\}, && \text{(6e)}\\
& \mathbf{s}_t^\top \mathbf{y}_t = 0, && \forall t \in \{k,k+1,\ldots,N-1\}, && \text{(6f)}\\
& \mathbf{c} + \bar{\mathbf{y}}_t = \mathbf{A}^\top \boldsymbol{\lambda}_t + \mathbf{s}_t, && \forall t \in \{k,k+1,\ldots,N-1\}, && \text{(6g)}\\
& \mathbf{y}_t, \mathbf{s}_t \in \mathbb{R}^n_+, \boldsymbol{\lambda}_t \in \mathbb{R}^{k+n_b}, && \forall t \in \{k,k+1,\ldots,N-1\}, && \text{(6h)}\\
& A_{1,2} \ge \cdots \ge A_{1,n}, && && \text{(6i)}
\end{aligned}$$

where $\bar{\mathbf{b}}_t^\top = \begin{bmatrix} x_{t-k+1} \cdots\ x_t\ \mathbf{b}^\top \end{bmatrix}$ and $\bar{\mathbf{y}}_t = \mathbf{y}_t - \begin{bmatrix} x_t \\ \mathbf{0}_{n-1} \end{bmatrix}$. To clarify the notation, $[n] = \{1,2,\ldots,n\}$ and $\mathbf{0}_{n-1}$ is a vector with $n-1$ zeroes. Moreover, since we only care for trainable parameters and the prediction for the training dataset, we denote $(\mathbf{A}^*, \mathbf{c}^*, \hat{\mathbf{x}}) = P_{train}(\mathbf{x},k,n,n_b)$.

The constraints (6b)–(6d) enforces our bound restrictions for the model parameters. Constraints (6e)–(6h) originate from the KKT conditions of each inner optimization problem. The last group of constraints (6i) are so-called symmetry-breaking constraints and are included to prevent multiple equivalent solutions. Without these constraints, we could reorder the variables $y_{t,2}, y_{t,3}, \ldots, y_{t,n}$ and the corresponding rows of $\mathbf{A}$ to get technically different solutions, but resulting in the same predictions. Eliminating such symmetries typically results in an easier problem.

Problem (6) is challenging and contains both complementarity constraints (6f) and nonconvex constraints with bilinear terms (6e) and (6g). The number

of nonconvex terms increases with the size of the inner problem (2). A copy of all these constraints is also generated by each data point used for training. The number of data points N, thus, strongly affects the tractability of problem (6). We do not detail how such problems can be solved, but they can be handed directly by MIP solvers such as Gurobi [15] and SCIP [6]. However, directly solving problem (6) is intractable even with relatively little data and small CQPB models. To handle problems of meaningful size and get a good, but potentially suboptimal solution, we use a two-stage strategy to solve the problem. First, we determine an initial solution by solving (6) with a small subset of the data. Then we fix all but one row of the $\mathbf{A}$ matrix and solve problem (6) with most elements of $\mathbf{A}$ fixed. Such a variables-fixing approach eliminates most of the bilinear terms. We then iterate by unlocking another row of $\mathbf{A}$ and fix all other rows in a block coordinate descent-type fashion. Algorithm 1 presents the complete procedure.

Algorithm 1 The two-stage alternating optimization-based training algorithm

Input: A time-series $\mathbf{x}$, a consecutive subset $\mathbf{x}_{init}$, the window size k, model complexity parameters n and n_b, the tolerance $\varepsilon \geq 0$, and time limit T_{max}.
Output: $\hat{\mathbf{A}}$, $\hat{\mathbf{c}}$, and $\hat{\mathbf{x}}$.

procedure Initialization Stage
  $(\mathbf{A}^0, \mathbf{c}^0, \hat{\mathbf{x}}_{init}) \leftarrow P_{train}(\mathbf{x}_{init}, k, n, n_b)$; ▷ Solve within T_{max}
end procedure
procedure Alternating Stage
  $j \leftarrow 1$, $\Delta \leftarrow +\infty$, $\text{OBJ}^0 \leftarrow +\infty$;
  while $\Delta \geq \epsilon$ **do**
    for $i = 1, \ldots, k + n_b$ **do** ▷ Fix all but the value of $\mathbf{A}_{i\cdot}$ with $\mathbf{A}^{j-1}$
      $(\mathbf{A}^j, \mathbf{c}^j, \hat{\mathbf{x}}) \leftarrow P_{train, \mathbf{A}_{s\cdot} = \mathbf{A}^{j-1}_{s\cdot}, \forall s \neq i}(\mathbf{x}, k, n, n_b)$; ▷ Solve within T_{max}
      $\text{OBJ}^j \leftarrow \sum_{t=k}^{N} (x_{t+1} - \hat{x}_{t+1})^2$;
      $\Delta \leftarrow \text{OBJ}^{j-1} - \text{OBJ}^j$;
      $j \leftarrow j + 1$; ▷ Count iterations
    end for
  end while
  $(\hat{\mathbf{A}}, \hat{\mathbf{c}}, \hat{\mathbf{x}}) \leftarrow (\mathbf{A}^j, \mathbf{c}^j, \hat{\mathbf{x}}^j)$.
end procedure

One can also initiate Algorithm 1 by forming an initial $\mathbf{A}$ matrix and $\mathbf{b}$ vector based on a trained AR model. The $\mathbf{A}$ matrix and $\mathbf{b}$ vector are then formed from the AR predictor similarly as in the proof of Proposition 1, and potentially filled out with zeroes if the QCPB predictor is of higher dimensionality. This provides a starting point with the same RMSE as the AR predictor on the training set.

4 Numerical Experiments

The implementation of data preprocessing, proposed methods, and other baseline approaches are carried out in Python. The optimization problems solved

in Algorithm 1 are solved by Gurobi 11.0.3 [15], and ARMA and RNNs, as baseline approaches, are implemented and trained by `Statsmodels` [23] and `PyTorch` [21], respectively. All the experiments were run on a laptop with an Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz processor and 16GB of RAM. To keep the training problems more manageable, we have forced bounds of ± 5 on dual variables (*i.e.*, λ and $\mathbf{s}$) for unbounded directions in problem (6). It is common to assume bounds on the dual variables when using the KKT reformulation, but one should be careful, as too restrictive bounds may lead to infeasibility. A more rigorous approach should be used to determine the bounds. However, here the goal has only been to investigate the potential of CQPB predictors.

In the initialization stage of Algorithm 1, we have used as many data points as possible, while keeping problem (6) solvable within a minute. The number of data points in the initialization varied between 5 and 13. We initially set a time limit T_{max} of 30 seconds for the subproblems, and we set T_{max} to 60 seconds when Gurobi could not find an initial solution in 30 seconds. To keep the problems simpler, we set $n = k + 1$ and $n_b = 0$, *i.e.*, one more variable in problem (2) than the number of data points used for prediction and the number of constraints. Preliminary results with $n = k+2$ did not show a clear difference in the models' predicting performance. For all experiments, we used $\varepsilon = 1e - 8$.

4.1 Data sets

We have considered the following data sets that vary in prediction difficulty:

1. **Nordpool** [20]: This dataset contains system prices for the Nordic energy market collected and published by the Nord Pool. We choose a consecutive sequence of daily prices measured at midnight from 2022-01-01 and ending with a given length. All the prices are scaled with a factor of 0.01.
2. **Global Temperature** [19]: This data set contains changes in the annual average global surface temperature, from which we use the data between 1909 and 2023. We specifically consider the yearly mean's locally weighted smoothing (lowess). The data points are shifted upward by the absolute value of the minimum value to make all values nonnegative.
3. **Sunspots** [26]: This dataset contains the monthly mean sunspot number from 1749-01-01 to 2017-08-31. The data are scaled by a factor of 0.01.

4.2 Reference models

To evaluate the predicting performance, we compared the following commonly used predictors as baselines:

– Autoregressive models (AR): Autoregressive models are created with the idea that the current value x_t can be explained as a function of the k past values $x_{t-1}, x_{t-2}, \ldots, x_{t-k}$. An autoregressive model of order k, abbreviated as AR(k), is defined as $x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_k x_{t-k} + w_t$, where $\phi_1, \phi_2, \ldots, \phi_k$ are constants and w_t is white noise with mean zero and variance σ_w^2. As AR models require processes to be (weakly) stationary, the

non-stationary time-series **Global Temperature** and **Sunspots** have been made stationary through first-order differencing [25].
– Recurrent neural networks (RNN): A simple RNN predicts the output at each time step depending on the current input and the hidden state from the previous time steps. They excel in simple tasks with short-term dependencies due to the ability to exploit the influence of previous inputs, such as time-series data [24, 16]. We chose small RNNs with one hidden layer of size 32 on each recurrent cell with a varying window size $m = \{2, 4, 6\}$. All the instances are trained with a batch size of 2 and 100 epochs, and the datasets are normalized in the preprocessing.

5 Results

First, we investigate what size of problem (6) is directly solvable with Gurobi. We tested two sizes for the CQPB model on the data set **Global Temperature**. We found that problem (6) could not be solved by Gurobi for either model under 10 minutes when using more than 6 and 8 data points for $k = 2$ and $k = 4$, respectively. This clearly shows the difficulty of directly optimizing (6).

To illustrate the simple training algorithm, we plot how the solution improves over iterations with a CQPB model with parameters $N = 50$, $N_{init} = 12$, and $k = 6$ for the data set **Nordpool** in Figure 1. Here Algorithm 1 ran 12 iterations taking 367s (with $\varepsilon = 1e - 8$). Algorithm 1 is a simple heuristic, but based on our experiments it is effective at producing good solutions relatively fast.

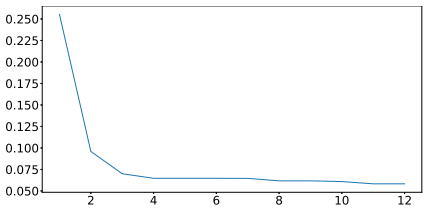

Fig. 1: The iterations of Algorithm 1 plotted against the MSE for the **Nordpool** data with $N = 50$ and $k = 6$.

5.1 Comparison with other predictive models

Table 1 presents the mean square error (MSE) for CQPB, AR(k), and RNN trained with the datasets and various parameters as well as the prediction mean square error (MSE_p) for the P one-step ahead predictions outside of the training data by the models. We have also included results with Algorithm 1 initiated with the AR solution, and this is refered to as QCPB-AR. Table 1 shows that CQPB overall performs similarly to both AR(k) and RNN. When it performs worse, it is not by a large margin and sometimes it is the best-performing model.

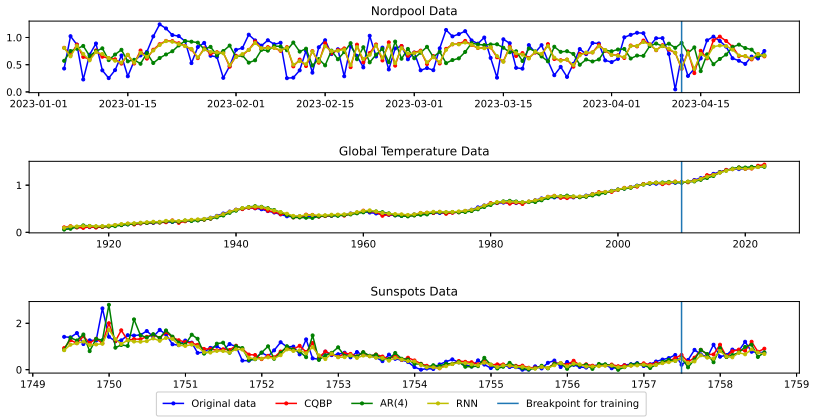

Fig. 2: Comparison of CQPB, AR(k), and RNN with $N = 100$, $P = 15$, and $k = 4$. The last 15 points show one-step predictions beyond the training data.

Figure 2 presents three plots comparing results from different models for each dataset separately, with $N = 100$, $N_{init} = 8$ (**Nordpool** and **Global Temperature**) or $N_{init} = 9$ (**Sunspots**), and $k = 4$ All three models show similar performance.

Whilst the performance of the CQPB predictor shows promise, the difficult training is clearly a drawback. For both AR and RNN, the training of the models is practically instantaneous, but the same cannot be said for the CQPB. The training times varied greatly and were highly dependent on the number of iterations to converge, with the fastest model taking just over a minute (two iterations with a 30 s limit each) and the slowest a bit over 15 minutes (15 iterations with a 60 s time limit). Most models took between 2 and 6 minutes to train.

From the experiments, it was clear that using the AR solution as an initialization was favorable. Otherwise the initialization in Algorithm 1 was sensitive with regards to the parameters, e.g., the number of data points used in the initialization. A more rigorous approach for determining bounds on the dual variables should also be investigated. Finally, we also want to mention that the performance of all models can most likely be improved by parameter tuning.

6 Conclusions

Overall, the CQPB predictor shows good capabilities for forecasting time-series data. Our simple heuristic training approach manages to compute good solutions, with predicting performance comparable to well-established methods for moderate-size datasets. However, the training process is still computationally challenging. Further research on CQPB predictors is, in our opinion, clearly motivated and needed to make them practically applicable.

	NORDPOOL DATA											
	$N = 50, P = 10$						$N = 100, P = 15$					
	$k = 2$		$k = 4$		$k = 6$		$k = 2$		$k = 4$		$k = 6$	
	MSE	MSE_p	MSE	MSE_p	MSE	MSE_p	MSE	MSE_p	MSE	MSE_p	MSE	MSE_p
CQPB	0.0649	0.0709	0.0608	0.0733	0.0582	0.0581	0.0571	0.0728	0.0548	0.0642	0.0549	0.0529
CQPB-AR	0.0630	0.0644	0.0592	0.0772	0.0533	0.0736	0.0560	0.0573	0.0533	0.0642	0.0508	0.0649
AR(k)	0.0649	0.0534	0.0608	0.0572	0.0521	0.0419	0.0570	0.0581	0.0544	0.0614	0.0503	0.0529
RNN	0.0640	0.0631	0.0604	0.0604	0.0540	0.0507	0.0608	0.0572	0.0592	0.0553	0.0543	0.0507
	GLOBAL TEMPERATURE DATA ($\times 10^{-2}$)											
	$N = 50, P = 10$						$N = 100, P = 15$					
	$k = 2$		$k = 4$		$k = 6$		$k = 2$		$k = 4$		$k = 6$	
	MSE	MSE_p	MSE	MSE_p	MSE	MSE_p	MSE	MSE_p	MSE	MSE_p	MSE	MSE_p
CQPB	0.0326	0.0316	0.0113	0.0236	0.0274	0.0387	0.0140	0.0216	0.0264	0.0350	0.0262	0.0227
CQPB-AR	0.0126	0.0833	0.0112	0.1525	0.0094	0.0318	0.0319	0.0285	0.0115	0.0752	0.0324	0.0268
AR(k)	0.0142	0.0192	0.0115	0.0164	0.0097	0.0173	0.0135	0.0138	0.0116	0.0146	0.0107	0.0161
RNN	0.0642	0.0617	0.0273	0.0263	0.0233	0.0224	0.0763	0.0757	0.0419	0.0396	0.0308	0.0355
	SUNSPOTS DATA											
	$N = 50, P = 10$						$N = 100, P = 15$					
	$k = 2$		$k = 4$		$k = 6$		$k = 2$		$k = 4$		$k = 6$	
	MSE	MSE_p	MSE	MSE_p	MSE	MSE_p	MSE	MSE_p	MSE	MSE_p	MSE	MSE_p
CQPB	0.1109	0.0616	0.1033	0.0431	0.1021	0.0405	0.0679	0.0953	0.0681	0.0796	0.1168	0.1043
CQPB-AR	0.1128	0.0614	0.1012	0.0425	0.1013	0.0507	0.0665	0.0944	0.0595	0.0899	0.0584	0.0895
AR(k)	0.1155	0.0212	0.1061	0.0251	0.1033	0.0261	0.0653	0.0641	0.0610	0.0718	0.0591	0.0766
RNN	0.1071	0.0999	0.0990	0.0931	0.0947	0.0911	0.0806	0.0818	0.0703	0.0711	0.0657	0.0682

Table 1: The performance of predictors trained with various parameters for the different data sets, given as mean squared error (MSE). MSE_p is the mean squared error for P one-step ahead predictions beyond the training data. For CQPB-AR, Algorithm 1 is initiated with the AR solution.

Acknowledgments. The authors gratefully acknowledge financial support from a grant by Göran Gustafsson Stiftelser to Jan Kronqvist, and support from Digital Futures at KTH.

Disclosure of Interests. The authors have no conflicts of interest.

References

1. Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., Kolter, J.Z.: Differentiable convex optimization layers. Advances in neural information processing systems **32** (2019)
2. Agrawal, A., Barratt, S., Boyd, S.: Learning convex optimization models. IEEE/CAA journal of automatica sinica **8**(8), 1355–1364 (2021)
3. Akaike, H.: Fitting autoregreesive models for prediction. In: Selected Papers of Hirotugu Akaike, pp. 131–135. Springer (1969)

4. Bard, J.F.: Practical bilevel optimization: algorithms and applications, vol. 30. Springer Science & Business Media (2013)
5. Biggar, D.R., Hesamzadeh, M.R.: The economics of electricity markets. John Wiley & Sons (2014)
6. Bolusani, S., Besançon, M., Bestuzheva, K., Chmiela, A., Dionísio, J., Donkiewicz, T., van Doornmalen, J., Eifler, L., Ghannam, M., Gleixner, A., et al.: The scip optimization suite 9.0. arXiv preprint arXiv:2402.17702 (2024)
7. Box, G.E., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: Time series analysis: forecasting and control. John Wiley & Sons (2015)
8. Breiman, L.: Random forests. Machine learning **45**, 5–32 (2001)
9. Chan, T.C.Y., Mahmood, R., Zhu, I.Y.: Inverse Optimization: Theory and Applications. Operations Research (2023)
10. Chan, T.C., Kaw, N.: Inverse optimization for the recovery of constraint parameters. European journal of operational research **282**(2), 415–427 (2020)
11. Dempe, S., Dutta, J.: Is bilevel programming a special case of a mathematical program with complementarity constraints? Mathematical Programming **131** (2012)
12. Frank, R.J., Davey, N., Hunt, S.P.: Time series prediction and neural networks. Journal of intelligent and robotic systems **31**, 91–103 (2001)
13. Gupta, R., Zhang, Q.: Decomposition and Adaptive Sampling for Data-Driven Inverse Linear Optimization. INFORMS Journal on Computing **34**(5) (2022)
14. Gupta, R., Zhang, Q.: Efficient learning of decision-making models: A penalty block coordinate descent algorithm for data-driven inverse optimization. Computers & chemical engineering **170**, 108123– (2023)
15. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2024), https://www.gurobi.com
16. Hewamalage, H.: Recurrent neural networks for time series forecasting: Current status and future directions. International Journal of Forecasting **37** (08 2020)
17. Katyal, C.: Differentiable convex optimization layers in neural architectures: Foundations and perspectives (2024), arXiv preprint
18. Kleinert, T., Labbé, M., Ljubić, I., Schmidt, M.: A survey on mixed-integer programming techniques in bilevel optimization. EURO Journal on Computational Optimization **9**, 100007 (2021)
19. NASA: Global surface temperature (Feb 2024), https://climate.nasa.gov/vital-signs/global-temperature/?intent=111
20. Nordpool: Day-ahead prices (2025), https://data.nordpoolgroup.com/auction/day-ahead/prices
21. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch. In: NIPS-W (2017)
22. Sapankevych, N.I., Sankar, R.: Time series prediction using support vector machines: a survey. IEEE computational intelligence magazine **4**(2), 24–38 (2009)
23. Seabold, S., Perktold, J.: statsmodels: Econometric and statistical modeling with python. In: 9th Python in Science Conference (2010)
24. Sherstinsky, A.: Fundamentals of recurrent neural network (RNN) and long short-term memory (lstm) network. Physica D: Nonlinear Phenomena **404** (2020)
25. Shumway, R.H., Stoffer, D.S.: Time Series Analysis and Its Applications. Springer Texts in Statistics, Springer New York, New York, NY, 1st ed. 2000. edn. (2000)
26. SIDC: Sunspots (Feb 2021), https://www.kaggle.com/datasets/robervalt/sunspots/data