IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 18 May 2024, accepted 7 July 2024, date of publication 29 July 2024, date of current version 12 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3434617

SURVEY

Predictive Analytics: An Optimization Perspective

LUIS RODRIGUES[1], (Senior Member, IEEE), AND SIDNEY N. GIVIGI[2], (Senior Member, IEEE)
[1]Department of Electrical and Computer Engineering, Concordia University, Montreal, QC H3G 1M8, Canada
[2]School of Computing, Queen's University, Kingston, ON K7K7B4, Canada

Corresponding author: Luis Rodrigues (luis.rodrigues@concordia.ca)

ABSTRACT Predictive analytics is concerned with making predictions of future outcomes based on past data using data statistics, machine learning, dynamic models and filtering algorithms. This paper provides a review of the theory of predictive analytics from an optimization perspective, including some new developments in convex optimization and quadratic neural networks. Several examples from many different areas show how the theory can be applied to analyze data and provide useful information to decision makers.

INDEX TERMS Classification, neural networks, predictive analytics, regression.

I. INTRODUCTION

The area of predictive analytics is concerned with making predictions of future outcomes based on past data using statistics, machine learning, dynamic models and filtering algorithms. Based on data analysis and dynamic models one can identify data patterns and determine any potential future impacts of the observed patterns. This can help policy developers make decisions regarding the evolution of safety-critical situations, risk assessment, or the identification of investment opportunities. In fact, predicting the future value of a dynamic process is of extreme importance in a multitude of different applications such as the prediction of stock values, gross domestic product, number of new cases in an epidemic, future weather, or the position and velocity of a moving vehicle, to name just a few. Predictive analytics has gathered increasing interest in the last decade [1]. Its development has been disseminated at all levels, including in student magazines [2]. Applications include studies of STEM student success [3], student retention [4], clinical diagnosis, diabetes, drug design, and multi-disease risks [5], [6], [7], [8], health informatics [9], analysis of big data in retailing [10], and finance [11], to name a few. Optimization is at the core of predictive analytics and it is a unifying approach among different predictive algorithms. However, there are very few contributions in the open literature that provide a general optimization framework for predictive analytics. Some references address application-specific modeling and optimization approaches for predictive analytics case studies [12], [13] whereas other references focus on specific aspects of optimization in predictive analytics [14], [15]. By contrast, this paper provides a general overview of predictive analytics from an optimization perspective with a focus on convex optimization. The paper will discuss the main areas of predictive analytics: regression, classification, clustering, and identification of dynamic models based on data.

Regression is concerned with finding an approximate curve that fits the available data and then use that curve for future predictions. When the curve is a straight line it is called linear regression. The most commonly used measure for assessing how well the curve fits the data is the sum of the squared errors, which leads to an optimization problem called least squares. Least squares has been analyzed in [16] where partial least-squares has been proposed to generate only a subset of the total number of predictors when one is dealing with very large amounts of data, so-called big-data. A cross-validated predictive test has been suggested in [17] for partial least squares to perform a parwise comparison of predictive power of two competing models.

Classification categorizes data into pre-defined different classes whereas clustering divides the data into different groups based on similar attributes. The different groups for clustering are not pre-defined as opposed to the case of classification. This is due to the different way in which learning from data is implemented in clustering when compared to classification. Classification uses supervised learning such as neural networks [18] and support vector

The associate editor coordinating the review of this manuscript and approving it for publication was Derek Abbott.

machines [19], whereas clustering uses unsupervised learning techniques such as k-means clustering and Loyd's algorithm. A comprehensive survey of k-means clustering including a thorough comparison of different versions of the k-means algorithm can be found in [20]. The stability of the centers of the k-means clusters is analyzed in [21]. An historical perspective of k-means clustering can be found in [22]

The identification of dynamic models can be divided into two different classes: input-output time series [23] and state space models [24]. State space models can be used to study the evolution of a collection of variables as a function of time. The variables are selected so that they completely determine the system's behaviour and are stacked together into what is called a state vector. Time series is concerned with the input-output description of a system as a function of time and is used for the identification of cycles and trends in time-stamped data.

This paper assumes that the reader has a standard background in calculus and linear algebra, and some previous exposure to time-series and dynamic models. Section II is dedicated to optimization problems, with an emphasis on convex optimization. This is followed by linear state space models in section III and linear time series input-output models in section IV. Neural network models are then addressed in section V. Regression is the first application to be described in section VI, followed by classification in section VII, system identification in section VIII, and clustering in section IX. The paper closes with the conclusions.

Notation: The sets of real numbers, real n vectors, and real $n \times n$ matrices are denoted by $\mathbb{R}$, $\mathbb{R}^n$, and $\mathbb{R}^{n\times n}$, respectively. In optimization problems the words "minimize in x" will be abbreviated to $\min_x$, "maximize in x" will be abbreviated by $\max_x$, and the words "subject to" will be represented by $s.t.$ The transpose of a matrix or vector will be denoted with a superscript T. For example x^T is the transpose of x. The partial derivative of a function $f(x)$, $x = [x_1 \ \dots \ x_n]^T$, with respect to x_i is denoted by $\partial_{x_i} f$. The norm-1 and norm-2 of a vector v are defined as $\|v\|_1 = \sum_i |v_i|$ and $\|v\|_2 = \sqrt{\sum_i v_i^2}$, respectively. The trace of a matrix $A \in \mathbb{R}^{n\times n}$ is the sum of its diagonal entries $tr(A) = \sum_{i=1}^n A_{ii}$. A matrix $P \in \mathbb{R}^{n\times n}$ is positive semi-definite (written $P \geq 0$) if all its eigenvalues are non-negative and positive definite (written $P > 0$) if all its eigenvalues are positive. A matrix $Q \in \mathbb{R}^{n\times n}$ is negative (semi-)definite if $-Q$ is positive (semi-)definite.

II. OPTIMIZATION

The unifying theme of all predictive analytics methods described in this paper is that they are formulated as an optimization problem as follows

$$\begin{aligned} \min_x \quad & f_0(x) \\ s.t. \quad & f_i(x) \leq 0, \ i = 1, \dots, m \\ & h_i(x) = 0, \ i = 1, \dots, p \end{aligned} \tag{1}$$

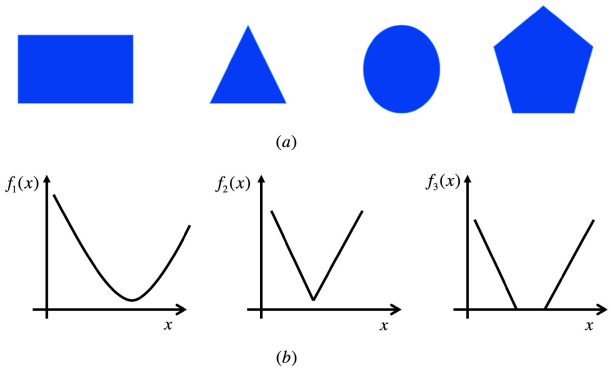

FIGURE 1. Convex sets (a) and convex functions (b).

where $f_i(x), i = 0, \dots m, h_i(x), i = 1, \dots, p$, are functions that will be assumed to be continuous everywhere and differentiable almost everywhere. The function $f_0(x)$ is called the objective function, which is what one is interested in minimizing, whereas the remainder functions $f_i(x)$ and $h_i(x)$ represent the inequality and equality constraints, respectively. When an optimization problem is unconstrained then it is stated as

$$\min_x \quad f_0(x). \tag{2}$$

For an unconstrained optimization with a differentiable objective function a necessary condition of optimality is

$$\partial_{x_i} f_0(x) = 0, \ i = 1, \dots, n. \tag{3}$$

For optimization problems with p affine equality constraints and without inequality constraints, one can solve the constraints first for p of the variables, and then replace these variables into the objective function leading to an unconstrained optimization problem in $n - p$ variables. For this new equivalent problem one can then use the necessary condition (3). An important subclass of optimization problems is the one corresponding to *convex optimization*. These problems can be solved efficiently both from a theoretical point of view (polynomial complexity) and from a practical point of view (available software packages). The following definitions characterize the general form of *convex optimization*.

Definition 1: A set $\mathcal{C}$ is convex if the line segment between any two points in $\mathcal{C}$ lies in $\mathcal{C}$, i.e, *if for any $x_1, \ x_2 \in \mathcal{C}$ and any $0 \leq \theta \leq 1$ we have*

$$\theta x_1 + (1 - \theta) x_2 \in \mathcal{C}.$$

□

Examples of convex sets are a circle in $\mathbb{R}^2$ including its interior (see figure 1a) or a sphere in $\mathbb{R}^3$ including also its interior. A set with the form of a horseshoe in $\mathbb{R}^2$ is not a convex set. When a set is not convex one can determine the smallest convex set that contains it, which is denoted by the convex hull of the original set.

Definition 2: The convex hull of a set $\mathcal{S}$ is the smallest convex set that contains $\mathcal{S}$. A simplex in $\mathbb{R}^n$ is defined as the convex hull of $n + 1$ affinely independent points. □

The affine independence in the definition guarantees that the simplex has a nonempty interior and does not vanish or collapse in some direction. Examples of simplices are intervals in $\mathbb{R}$, triangles in $\mathbb{R}^2$ (see figure 1a) and tetrahedrons in $\mathbb{R}^3$. The epigraph of a function is the region lying above the graph of the function, including the graph itself. Convex functions are functions whose epigraph is a convex set. Some examples are an exponential function and a quadratic polynomial with positive second derivative (see figure 1b). Note that convex functions do not need to be differentiable everywhere. Another definition of convex function is as follows.

Definition 3: A function f is convex if the domain of f $\left(Dom_f\right)$ is a convex set and if the line segment between $(x, f(x))$ and $(y, f(y))$ lies above the graph of f, i.e, *if for all $x,\ y \in Dom_f$ and $0 \leq \theta \leq 1$ we have*

$$f\left(\theta x + (1-\theta)y\right) \leq \theta f(x) + (1-\theta) f(y).$$

A function f is concave if $-f$ is convex. □

A general *convex optimization program* has the form (1) where $f_0, \ldots, f_m$ are convex functions and $h_i(x) = a_i^T x - b_i,\ i = 1, \ldots, p$, are affine.

The next four sections will introduce examples of predictive analytics optimization problems.

A. LEAST SQUARES

The first approach when analyzing data is quite often to try to fit a curve to the data so that future predictions can be made using this curve. To that aim, we assume that we have a linear model for the vector of data measurements y as a function of some unknown vector θ with an additive error e, i.e.,

$$y = H\theta + e. \tag{4}$$

If one wishes to fit a line to data pairs $(x_i, y_i), i = 1, \ldots, N$ then the model (4) is related to the equation of the line as shown in the linear regression example 1. We wish to find θ that provides a small estimation error $e = y - H\theta$ while at the same time not having a very large magnitude $\|\theta\|_2$. The intuition behind this idea is that a solution θ with a large magnitude may be associated with overfitting the model. To solve this problem one can define an objective function with a sum of two terms: one that weights the magnitude of the error and another that weights the magnitude of the estimated variable. When both terms are quadratic, this is a regularized least squares optimization that can be formulated as

$$\begin{aligned} \min_{\theta, e} \quad & e^T L e + \theta^T M \theta \\ \text{s.t.} \quad & y = H\theta + e \end{aligned} \tag{5}$$

for given positive definite symmetric matrices L and M that are selected by the designer. Note that there is a quadratic term $\theta^T M \theta$ in the objective function of (5) that penalizes large values of the entries of θ. If $M = I$ then the extra term is $\theta^T\theta$, which is $\|\theta\|_2^2$. For diagonal positive definite matrices M the extra term is a weighted sum of squares. The same interpretation can be used for the parameter L with respect to e. The equality constraint of the optimization problem (5) can be solved for e as

$$e = y - H\theta. \tag{6}$$

Replacing (6) into the objective function and rearranging using the rules

$$\begin{aligned} (A+B)^T &= A^T + B^T, \\ (AB)^T &= B^T A^T, \end{aligned} \tag{7}$$

and using the fact that $y^T L H\theta = \theta^T H^T L y$ because it is a scalar and $L = L^T$, one arrives at the equivalent optimization

$$\min_{\theta} \quad f(\theta), \tag{8}$$

where

$$f(\theta) = \theta^T \left(H^T L H + M\right)\theta - 2\theta^T H^T L y + y^T L y. \tag{9}$$

To find the solution of the minimization (8) we solve the necessary condition

$$\partial_\theta f = 2\left(H^T L H + M\right)\theta - 2H^T L y = 0. \tag{10}$$

The matrix $\left(H^T L H + M\right)$ is always invertible for a positive definite M, and the solution of (10) is the regularized least squares estimate

$$\hat{\theta} = (H^T L H + M)^{-1} H^T L y. \tag{11}$$

If M was allowed to be positive semi-definite (instead of positive definite) then making $M = 0$ one would find the weighted least squares solution, which assumes H to be maximum rank so that $H^T L H$ is invertible.

B. K-MEANS CLUSTERING

Data clustering belongs to the class of unsupervised learning because the data is not assumed to be labeled. One assumes that there is a collection of data vector samples $v_i \in \mathbb{R}^n, i = 1, \ldots, N$, and that we are given a number of k clusters in which to split the data points. The objective is then to find the regions $\mathcal{R}_i, i = 1, \ldots, k$, partitioning $\mathbb{R}^n$ that correspond to the clusters. After finding these regions, any data point can be allocated to one of the k clusters $\mathcal{R}_i$. A data point is allocated to the cluster $\mathcal{R}_i$ if it is closer to the centroid (mean) x_i of that cluster than to any other centroid $x_j, j \neq i$. This yields what is called a Voronoi partition of $\mathbb{R}^n$. The regions $\mathcal{R}_i$ in this partition are thus defined as

$$\mathcal{R}_i = \{x \mid \|x - x_i\|_2 < \|x - x_j\|_2\ \forall j \neq i\} \tag{12}$$

for $i = 1, \ldots, k$. Note that if any data point is equidistant from two or more centers then it will not be classified in any cluster. In practice when that happens the data point is arbitrarily allocated to one of the regions corresponding to the centers from which it is equidistant. Given k and $V =$

$[v_1 \dots v_N]^T$, the optimization for the k-means clustering is

$$\min_{x_i} \quad \sum_{i=1}^{k}\sum_{j=1}^{N} a_{ij}\|v_j - x_i\|_2^2$$
$$\text{s.t.} \quad (12), \quad a_{ij} = \begin{cases} 0, & v_j \notin \mathcal{R}_i \\ 1, & v_j \in \mathcal{R}_i \end{cases}. \qquad (13)$$

Unfortunately, this problem is not convex. In fact it has been shown to be *NP* hard, which makes it computationally intractable. Therefore, one resorts to heuristics to find approximate solutions of the k-means clustering problem. The most widely used heuristic is Loyd's algorithm (see table).

Algorithm 1 Loyd's Algorithm

```
function kmeans (V, k)  [V ∈ R^{N×n} and k ∈ R]
    Initialize k × n array Centers at random
    Initialize k × n array PreviousCenters with zeros
    Initialize Cluster with k void n-arrays
    while norm(Centers - PreviousCenters) > ϵ do
        for j = 1, ..., N do
            for i = 1, ..., k do
                d(i) = norm(Centers(i, :) − V(j, :))
            end
            [ClosestCenter, m] = min(d)
            Cluster{m} = [Cluster{m}; V(j, :)]
        end
        PreviousCenters = Centers
        for i = 1, ..., k do
            Centers(i, :) = MeanRow(Cluster{i})
        end
    end return Centers
```

C. SUPPORT VECTOR MACHINES

As we have seen in the previous section, clustering is unsupervised learning because there are no labels to the training data points allocating them to a specific cluster. In classification problems we assume that the different clusters (called classes) have training data with the corresponding class label. Therefore, the classifier is obtained by supervised training. A support vector machine is a classifier that is applicable to data that can be separated by a hyperplane called a separating hyperplane. The data is said to be linearly separable if such a hyperplane can be found. The objective is to find among the set of all hyperplanes that are able to separate two different classes of data, the one that maximizes the minimum distance of the hyperplane to the closest data point. This minimum distance $d_{\min}$ is called the hyperplane margin. After the hyperplane is determined, one class of data points will lie above the hyperplane whereas the other class will lie below the hyperplane (see figure 2). This idea can be extended to the classification of several different classes of data. One way to do this is to find first one hyperplane for each pair of classes separately, and then to combine all hyperplanes in a single separating surface.

In order to train a support vector machine by solving an optimization problem let us first define the affine function

$$f(x) = w^T x + w_0. \qquad (14)$$

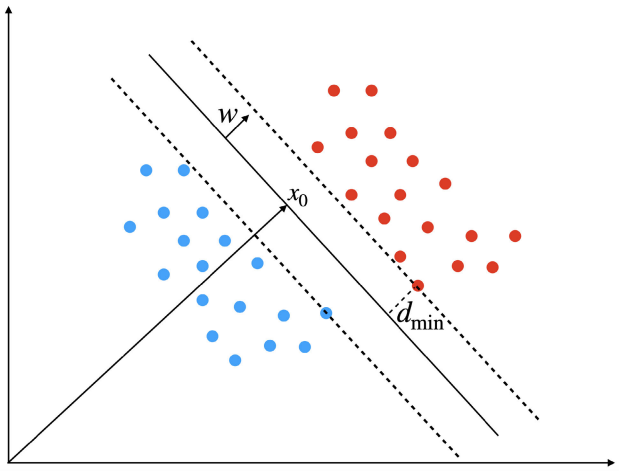

FIGURE 2. Support vector machine hyperplane.

The equation of the separating hyperplane is $f(x) = 0$. We label the two different classes of data points by defining a function $y : \mathbb{R}^n \to \{-1, 1\}$. For the data points $x \in \mathbb{R}^n$ located above the hyperplane (red points in figure 2) one has $f(x) > 0$ and $y(x) = 1$. The data points x located below the hyperplane (blue points in figure 2) will satisfy $f(x) < 0$ and $y(x) = -1$. Therefore, all data points satisfy the constraint

$$|f(x^i)| = y(x^i)\left(w^T x^i + w_0\right), \ i = 1, \dots, N. \qquad (15)$$

The vector x_0 (see figure 2) that is parallel to the vector w and connects the origin to the hyperplane with equation $f(x) = 0$ can be written as

$$x_0 = \alpha \frac{w}{\|w\|_2} \qquad (16)$$

and must satisfy the equation

$$f(x_0) = w^T x_0 + w_0 = 0. \qquad (17)$$

Therefore, replacing (16) into (17) and solving for α, one concludes that

$$x_0 = -\frac{w_0 w}{\|w\|_2^2}. \qquad (18)$$

To be able to determine $d_{\min}$ in figure 2 we must first compute the orthogonal distance of any vector x^i to the hyperplane with equation $f(x) = 0$. This distance is equal to the magnitude of the orthogonal projection of $x^i - x_0$ onto w computed as

$$d(x^i) = \left|\frac{w^T}{\|w\|_2}\left(x^i - x_0\right)\right| = \frac{|f(x^i)|}{\|w\|_2} = \frac{y(x^i)f(x^i)}{\|w\|_2}, \qquad (19)$$

where equation (15) was used to remove the absolute value. Given N data points x^i, the problem of finding the hyperplane that maximizes the hyperplane margin (i.e., the minimum distance $d_{\min}$) can then be formulated as

$$\max_{w, w_0, \gamma} \quad \gamma$$
$$\text{s.t.} \quad \frac{y(x^i)f(x^i)}{\|w\|_2} \geq \gamma, \ i = 1, \dots, N. \qquad (20)$$

To be able to separate the data one must find the vector w with the right orientation (see figure 2). Note that one can set without loss of generality the constraint $\gamma\|w\|_2 = 1$, given that it has no effect on the orientation of the vector w. Adding the constraint $\|w\|_2\gamma = 1$ to the optimization (20), it becomes the equivalent problem

$$\begin{aligned} &\min_{w,w_0} \quad \|w\|_2^2 \\ &\text{s.t.} \quad y(x^i)f(x^i) \geq 1, \ i = 1, \ldots, N. \end{aligned} \tag{21}$$

Since the data is never perfect and in certain cases not all data points may be separated by a hyperplane, one can relax the constraints of the problem (21) to allow for some data points to be misclassified. This must be done while still making sure that the minimum amount of misclassified points is desired. The way to proceed is to add what are called slack variables $\zeta_i \geq 0, \ i = 1, \ldots, N$, which are all stacked into a vector ζ. Similarly to the case of least squares in section II-A, the objective function is changed in order to be a trade off between two objectives: minimizing the norm-2 of $\|w\|_2$ and minimizing the norm-1 of ζ. The trade-off will depend on a parameter $\beta \geq 0$ that is called a regularization parameter. The modified optimization program is known as the soft margin formulation and can be stated as

$$\begin{aligned} &\min_{w,w_0} \quad \|w\|_2^2 + \beta \sum_{i=1}^{N} \zeta_i \\ &\text{s.t.} \quad y(x^i)f(x^i) \geq 1 - \zeta_i, \ \zeta_i \geq 0, \ i = 1, \ldots, N. \end{aligned} \tag{22}$$

Note that if $0 < \zeta_i < 1$ the data point x^i is classified correctly but it does not lie above the minimum margin. If $\zeta_i > 1$ then the data point x^i is misclassified.

D. NEURAL NETWORKS

Feedforward neural networks map a given input vector x into an output vector $\hat{y}$ and are composed of different layers. The last layer performs an affine combination of its inputs. All layers but the last are called hidden layers and each hidden layer has several neurons. When presented with an input vector s with elements $s_i, \ i = 1, \ldots, n$, a neuron output is

$$z(s) = \sigma\left(w^T s + w_0\right) \tag{23}$$

where w and w_0 form the vector of weights of the neuron and σ is a nonlinear activation function that models the "firing" characteristics of the neuron. This function is commonly chosen to be a sigmoid (e.g., tangent hyperbolic), or $ReLU(z) = \max(0, z)$, or a polynomial. Note that by creating an augmened input vector $\bar{s} = [s^T \ \ 1]^T$ and an augmented weight vector $\bar{w} = [w^T \ \ w_0]^T$ one can rewrite equation (23) as $z(\bar{s}) = \sigma\left(\bar{w}^T\bar{s}\right)$. Therefore, without loss of generality, the affine terms w_0 will not be considered.

Figure 3 shows an example of a neural network with a single output and one hidden layer. The network is called a quadratic neural network when the activation functions σ are quadratic polynomials. These networks have the advantage

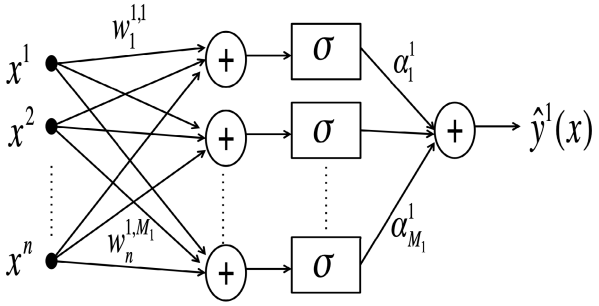

FIGURE 3. Single output two-layer neural network.

that training can be performed by a convex optimization and will be the focus of this section. The structure of a quadratic feedforward neural network with one output, as presented in figure 3, was suggested in [25]. Although quadratic neural networks can have several outputs, the network proposed in [25] is constrained to have a single hidden layer with $M = \sum_{k=1}^{p} M_k$ neurons. Each output is connected to M_k neurons in the hidden layer. The value of each network output $k = 1, \ldots, p$, is

$$\hat{y}^k(x) = \hat{f}^k(x) = \sum_{j=1}^{M_k} \sigma\left(x^T w^{k,j}\right) \alpha_j^k \tag{24}$$

where the activation function is quadratic and is written as

$$\sigma(z) = az^2 + bz + c \tag{25}$$

where $a \neq 0, b, c$, are pre-defined constants that parameterize the quadratic activation function. The weights w connect the input $x \in \mathbb{R}^n$ to each neuron in the hidden layer (figure 3).

The notation of equation (24) where the weights leaving the input are denoted by w and the weights leaving the hidden layer are denoted by α will be used throughout the paper. For the weights we will use superscript indices to indicate the target and subscript indices to indicate the source of each connection. For example, to denote the weight connecting the first input component to the second hidden neuron of the third output we use $w_1^{3,2}$. The desired (label) outputs will be denoted by y and the actual outputs of the network will be denoted by $\hat{y}$.

As already mentioned, one of the advantages of quadratic neural networks is that the training of the weights can be done by solving a convex optimization problem, which guarantees convergence to the global optimum. To do this, following [25] it will be assumed that the weights $w^{k,j}$ are normalized to have unit 2-norm. Additionally, a regularization term on the 1-norm of α_j will be included in the objective function for the network training. Furthermore, we assume the more general case $w^{k,j} = w^j$ of all M neurons connected to each output $\hat{y}^k$. Using a loss (error) function $l(\cdot)$ the original (primal) non-convex training problem for a quadratic neural network is [25]

$$\begin{aligned} &\min_{w^j,\alpha_j} \ l(\hat{y} - y) + \beta \sum_{i=1}^{M} \|\alpha_i\|_1 \\ &\text{s.t.} \quad \hat{y}^k = \sum_{j=1}^{M} \sigma\left(x^T w^j\right) \alpha_j^k, \end{aligned}$$

$$\|w^j\|_2 = 1,\ k = 1, \ldots, p,\ j = 1, \ldots, M, \tag{26}$$

for fixed $a \neq 0, b, c$, and a fixed regularization coefficient $\beta \geq 0$. Following [25], unless otherwise stated, the parameters a, b, c, will be selected to approximate in the least squares sense the function $ReLU(z) = \max(0, z)$ in the interval $[-5, 5]$ and are equal to $a = 0.0937, b = 0.5, c = 0.4688$. The following result from [25] recasts the training as an equivalent convex optimization problem.

Lemma 1: [25] Given fixed $a \neq 0, b, c$, and a fixed regularization coefficient $\beta \geq 0$, the solution of the convex problem that is dual to (26) and is formulated as

$$\begin{aligned}
&\min_{Z_+^k,\, Z_-^k} l(\hat{y} - y) + \beta \sum_{k=1}^{p} \left(Z_+^{k,4} + Z_-^{k,4} \right) \\
&s.t. \quad \hat{y}_i^k = \bar{x}_i^T \begin{bmatrix} a\left(Z_+^{k,1} - Z_-^{k,1}\right) & \frac{b}{2}\left(Z_+^{k,2} - Z_-^{k,2}\right) \\ \frac{b}{2}\left(Z_+^{k,2} - Z_-^{k,2}\right)^T & c\, tr\left(Z_+^{k,1} - Z_-^{k,1}\right) \end{bmatrix} \bar{x}_i \\
&\quad Z_+^{k,4} = tr\left(Z_+^{k,1}\right), Z_-^{k,4} = tr\left(Z_-^{k,1}\right), \\
&\quad Z_+^k = \begin{bmatrix} Z_+^{k,1} & Z_+^{k,2} \\ \left(Z_+^{k,2}\right)^T & Z_+^{k,4} \end{bmatrix}, \\
&\quad Z_-^k = \begin{bmatrix} Z_-^{k,1} & Z_-^{k,2} \\ \left(Z_-^{k,2}\right)^T & Z_-^{k,4} \end{bmatrix}, \\
&\quad Z_+^k \geq 0,\ Z_-^k \geq 0,\ \bar{x}_i^T = [x_i^T \ \ 1], \\
&\quad k = 1, \ldots, p,\ i = 1, \ldots N,
\end{aligned} \tag{27}$$

where $l(\cdot)$ is a convex loss function, provides a global optimal solution for the parameters $Z_+^k, Z_-^k \in \mathbb{R}^{(n+1)\times(n+1)}$, for $k = 1, \ldots, p$, when the number of neurons satisfies $M \geq M_*$ with

$$M_* = \sum_{k=1}^{p} \left[rank\left(Z_+^{k*}\right) + rank\left(Z_-^{k*}\right) \right], \tag{28}$$

where Z_+^{k*} and Z_-^{k*} for $k = 1, \ldots, p$, are the solution of the optimization problem (27) given N input data vectors $x_i \in \mathbb{R}^n$ with corresponding labels $y_i \in \mathbb{R}^p$ and $\hat{y}^k$ is given by (30). Moreover, the optimal value of the solutions of problems (26) and (27) are the same and the duality gap is zero. □

To determine the weights of the neural network, a clear advantage of quadratic neural networks with a single hidden layer is that the optimal network weights and architecture can be extracted from Z_+^{k*} and Z_-^{k*} using Algorithm 2 from [25] where

$$G = \begin{bmatrix} I_n & 0 \\ 0 & -1 \end{bmatrix} \tag{29}$$

and I_n is the identity matrix of order n. Instead of describing the neural network by its weights, in this paper we will focus on the description of the neural network as a quadratic form because such a description leads to an analytical expression, namely the one in equation (30) where $\bar{x} = [x^T \ \ 1]^T$. The quadratic form (30) is equivalent to the quadratic form in (27) and to expression (24) (see [25]).

$$\hat{y}^k = \hat{f}^k(x) = \bar{x}^T \begin{bmatrix} aZ_1^k & \frac{b}{2}Z_2^k \\ \frac{b}{2}\left(Z_2^k\right)^T & ctr\left(Z_1^k\right) \end{bmatrix} \bar{x} = \bar{x}^T \bar{Z}^k \bar{x} \tag{30}$$

Algorithm 2 Neural Decomposition With Tolerance **tol** [25]

function NeuralDecomposition ($\mathbf{Z}^*$, **tol**)
  Compute rank-1 decomposition $\mathbf{Z}^* = \sum_{j=1}^{r} v_j v_j^T$ using eigenvector decomposition keeping all eigenvectors with eigenvalues $\lambda >$ **tol**.
  Create a list of vectors $\boldsymbol{v}\ \mathcal{D}\ \{\boldsymbol{v_1}, \ldots, \boldsymbol{v_r}\}$ and a void output list $\mathbf{V}$
  for $k = 1, \ldots, r-1$ **do**
    $v_1 = v_k$
    if $v_1^T G v_1 = 0$ **then** $\mathbf{v}^* = v_1$
    else find $j \in [k+1, r]$: $\left(v_1^T G v_1\right)\left(v_j^T G v_j\right) < 0$ and set $\mathbf{v}^* = \frac{v_1 + \gamma v_j}{\sqrt{1+\gamma^2}}$ where $\gamma = \frac{-2v_1^T G v_j + \sqrt{\Delta}}{2v_j^T G v_j}$ and $\Delta = 4\left[\left(v_1^T G v_j\right)^2 - \left(v_1^T G v_1\right)\left(v_j^T G v_j\right)\right]$
    Remove v_k from list of vectors v and insert $v^* = \frac{v_j - \gamma v_1}{\sqrt{1+\gamma^2}}$ at the end of the list.
    Add $\mathbf{v}^*$ to the list $\boldsymbol{V}$
  Add last element of list $\boldsymbol{v}$ to list $\boldsymbol{V}$ and **return** $\boldsymbol{V}$

Algorithm 2 takes as inputs $Z^* = Z_+^{k*}$ and $Z^* = Z_-^{k*}$ for $k = 1, \ldots, p$, which are the solution of the optimization problem (27) and thus satisfy $tr\,(Z^*G) = 0$ [25]. At the end of the algorithm there will be two lists of vectors for each value of $k = 1, \ldots, p$. One will be the list $\{v_+^{k,1}, \ldots, v_+^{k,r_k^+}\}$ for $Z^* = Z_+^{k*}$ and the other will be the list $\{v_-^{k,1}, \ldots, v_-^{k,r_k^-}\}$ for $Z^* = Z_-^{k*}$ where

$$\begin{aligned}
v_+^{k,j} = \begin{bmatrix} c_+^{k,j} \\ d_+^{k,j} \end{bmatrix}, \quad v_-^{k,j} = \begin{bmatrix} c_-^{k,j} \\ d_-^{k,j} \end{bmatrix}, \\
c_+^{k,j}, c_-^{k,j} \in \mathbb{R}^n, \quad d_+^{k,j}, d_-^{k,j} \in \mathbb{R}.
\end{aligned} \tag{31}$$

For each output $k = 1, \ldots, p$, the weights in the first layer of the neural network will be $(w^{k,1^+}, \ldots, w^{k,r_k^+}, w^{k,1^-}, \ldots, w^{k,r_k^-})$ where [25]

$$w^{k,j^+} = \frac{c_+^{k,j}}{\|c_+^{k,j}\|_2}, \quad w^{k,j^-} = \frac{c_-^{k,j}}{\|c_-^{k,j}\|_2}. \tag{32}$$

The weights of the second layer of the neural network will be

$$(\alpha_1^{k^+}, \ldots, \alpha_{r_k^+}^{k^+}, \alpha_1^{k^-}, \ldots, \alpha_{r_k^-}^{k^-})$$

where [25]

$$\alpha_j^{k^+} = \left(d_+^{k,j}\right)^2, \quad \alpha_j^{k^-} = -\left(d_-^{k,j}\right)^2. \tag{33}$$

The output k of the neural network for the input x is given by

$$\hat{y}^k(x) = \hat{f}^k(x) = \sum_{j=1}^{r_k^+} \sigma\left(x^T w^{k,j^+}\right) \alpha_j^{k^+} + \sum_{j=1}^{r_k^-} \sigma\left(x^T w^{k,j^-}\right) \alpha_j^{k^-}. \tag{34}$$

Renumbering the weights w^{k,j^-} and $\alpha_j^{k^-}$ to run from $j = r_k^+ + 1$ to $j = r_k^+ + r_k^-$ we can rewrite (34) as (24) with $M_k = r_k^+ + r_k^-$.

Remark 1: The convex formulation of the neural network training (27) followed by the neural decomposition algorithm 2 yields a network architecture in which any given hidden layer neuron is only connected to one output (see figure 3 for the output $\hat{y}^1$). This is in contrast to the primal training problem (26) which assumed the general case of all hidden layer neurons connected to all outputs. It is proved in [25] that the solution of the convex training problem yields the same optimal value for the objective function as the primal training problem if $M \geq M_*$ with M_* given by (28). Additionally, it is proved in [26] that the convex problem (27) is equivalent to a least squares problem when $l(y - \hat{y}) = \|y - \hat{y}\|_2^2$ and $\beta = 0$, leading to the analytical solution (11) with $L = I$ and $M = 0$.

III. LINEAR STATE SPACE MODELS

State space models can be used to study the evolution of a collection of variables as a function of time. The variables are selected so that they completely determine the system's behaviour and are stacked together into what is called a state vector. Perhaps the simplest state space model that is familiar to everyone is the model of a bank account, which is

$$x(t+1) = x(t) + rx(t) - d(t) + u(t), \tag{35}$$

where $x(t)$ is the balance in the account at the end of month t and represents the state of the account, r is the monthly interest rate, $d(t)$ represents the monthly expenses, and $u(t)$ represents the monthly deposits in the account. This model can be rewritten as

$$x(t+1) = Ax(t) + Bu(t) + Wd(t), \tag{36}$$

with $A = 1 + r, B = 1, W = -1$. The description in equation (36) is valid for any discrete-time linear model with $x \in \mathbb{R}^n, u \in \mathbb{R}^m, d \in \mathbb{R}^l$ where $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}, W \in \mathbb{R}^{n\times l}$ are the parameters. The variables $x(t), u(t), d(t)$ are called the state, the input, and the disturbance, respectively. Each discrete-time linear model will have its own matrices A, B, W but any discrete-time linear model can be represented in the form (36). Starting from an initial condition $x(0)$ and knowing the values of $u(0)$ and $d(0)$ one can compute the value of $x(1)$ using equation (36) as

$$x(1) = Ax(0) + Bu(0) + Wd(0).$$

From the knowledge of $x(1), u(1), d(1)$ one can then compute $x(2)$ as

$$\begin{aligned} x(2) &= Ax(1) + Bu(1) + Wd(1) \\ &= A^2x(0) + ABu(0) \\ &\quad + AWd(0) + Bu(1) + Wd(1). \end{aligned}$$

Continuing this procedure one can find the general solution of equation (36) for any time t to be

$$x(t) = A^t x(0) + \sum_{i=0}^{t-1} A^i \left[Bu(t-1-i) + Wd(t-1-i)\right]. \tag{37}$$

In many applications the variable $u(t)$ corresponds to a decision or control action and the variable $d(t)$ corresponds to a disturbance or noise signal, which is typically not known. Therefore, when using data to determine an estimate of the model parameters we will restrict ourselves to the case

$$x(t+1) = Ax(t) + Bu(t) + e(t), \tag{38}$$

where $x(t), u(t)$ are assumed to be measured at M different times and $e(t)$ is the model error. To find the parameters A, B that minimize the sum of the errors squared we define the objective function

$$f_0(A, B) = \sum_{k=0}^{M-1} e^T(k) e(k). \tag{39}$$

Using equation (38) and the trace operator this function can be rewritten as

$$\begin{aligned} f_0(A, B) = \sum_{k=0}^{M-1} tr[&(x(k+1) - Ax(k) - Bu(k)) \\ &(x(k+1) - Ax(k) - Bu(k))^T]. \end{aligned}$$

The optimization problem to be solved to find the parameters A, B is then

$$\min_{A,B} \quad f_0(A, B). \tag{40}$$

Using the results

$$\partial_A tr\left[AM^T\right] = M, \tag{41}$$

$$\partial_A tr\left[AMA^T\right] = 2AM, \tag{42}$$

and solving the necessary conditions of optimality

$$\partial_A f_0(A, B) = 0, \tag{43}$$

$$\partial_B f_0(A, B) = 0, \tag{44}$$

leads to

$$B = \left(R_{x^+u} - R_{x^+x} R_{xx}^{-1} R_{ux}^T\right) \left(R_{uu} - R_{ux} R_{xx}^{-1} R_{ux}^T\right)^{-1}, \tag{45}$$

$$A = \left(R_{x^+x} - BR_{ux}\right) R_{xx}^{-1}, \tag{46}$$

with

$$R_{ab} = \frac{1}{M}\sum_{k=0}^{M-1} a(k)b^T(k), \tag{47}$$

where $a(k)$ and $b(k)$ can be any of the variables $x(k)$, $x^+(k) = x(k+1)$, $u(k)$. The solution (45)–(46) is called the maximum likelihood estimation of the parameters A and B and it assumes that the data is "rich enough" so that all of the inverse matrices in the formulas (45)–(46) exist (sometimes called the persistent excitation assumption [23]). Note that for unforced systems one has $B = 0$ and (46) becomes

$$A = R_{x^+x}R_{xx}^{-1}. \tag{48}$$

IV. LINEAR INPUT-OUTPUT TIME SERIES MODELS

Sometimes one cannot measure the whole state of a system and the only available data is a time series of a scalar output of the system $y(t)$. In that case, instead of building linear state space models one can turn to linear parametric input-output time series models. The most commonly used models are autoregressive (AR), moving average (MA), autoregressive exogenous (ARX), and autoregressive moving average exogenous (ARMAX). To be able to formulate such models one must first define the delay operator

$$q^{-1}y(t) = y(t-1). \tag{49}$$

The input-output ARMAX model can be written as

$$A(q^{-1})y(t) = B(q^{-1})u(t-d) + C(q^{-1})e(t), \tag{50}$$

where

$$A(q^{-1}) = 1 + a_1q^{-1} + \ldots + a_{n_a}q^{-n_a}, \tag{51}$$
$$B(q^{-1}) = b_0 + b_1q^{-1} + \ldots + b_{n_b}q^{-n_b}, \tag{52}$$
$$C(q^{-1}) = 1 + c_1q^{-1} + \ldots + c_{n_a}q^{-n_a}, \tag{53}$$

and where $y(t)$, $u(t)$, $e(t)$ represent the output, (exogenous) input, and error (or noise) signals, respectively. The order of the polynomials $A(q^{-1})$, $B(q^{-1})$ and $C(q^{-1})$ are n_a, n_b and n_a, respectively. The delay of the system is the difference of the number of delays in the output and input, $d = n_a - n_b > 0$.

A moving average (MA) model describes the output $y(t)$ as a weighted sum of present and past error (noise) values and is written as

$$y(t) = C(q^{-1})e(t), \tag{54}$$

or equivalently as

$$y(t) = e(t) + c_1e(t-1) + \ldots + c_{n_a}e(t-n_a). \tag{55}$$

In an autoregressive (AR) model the variable $y(t)$ is a function of its past values (autoregresses on itself) and of the current value of the error signal. Therefore, the AR model is written as

$$A(q^{-1})y(t) = e(t), \tag{56}$$

or equivalently as

$$y(t) = -a_1y(t-1) - \ldots - a_{n_a}y(t-n_a) + e(t). \tag{57}$$

The autoregressive exogenous (ARX) model is similar to the autoregressive model but it also includes terms in an exogenous input signal $u(t)$. It is written as

$$A(q^{-1})y(t) = B(q^{-1})u(t-d) + e(t), \tag{58}$$

or equivalently as

$$\begin{aligned} y(t) = &-a_1y(t-1) - \ldots - a_{n_a}y(t-n_a) \\ &+ b_0u(t-d) + \ldots + b_{n_b}u(t-d-n_b) \\ &+ e(t). \end{aligned} \tag{59}$$

This model can be rewritten in matrix form for a data set consisting of time samples $t = n_a, \ldots, n_a + N - 1$. It can be cast as in (4) where $\theta = [a_1, \ldots, a_{n_a}, b_0, \ldots, b_{n_b}]^T$ and

$$H_1 = \begin{bmatrix} -y(n_a-1) & \ldots & -y(0) \\ \vdots & \vdots & \vdots \\ -y(n_a+N-2) & \ldots & -y(N-1) \end{bmatrix}$$
$$H_2 = \begin{bmatrix} u(n_b) & \ldots & u(0) \\ \vdots & \vdots & \vdots \\ u(n_b+N-1) & \ldots & u(N-1) \end{bmatrix}$$
$$H = \begin{bmatrix} H_1 & H_2 \end{bmatrix}, y = [y(n_a) \ldots y(n_a+N-1)].^T \tag{60}$$

It is assumed that $N \geq n_a + n_b + 1$. The least squares estimation of the parameter vector θ can then follow the procedure outlined in section II-A.

V. NEURAL NETWORK MODELS

Neural networks can also be used to obtain input-output and state space models of a discrete-time dynamic system. It is assumed that a collection of input-output data pairs $\{u(k), y(k)\}_{k=1}^N$ are measured. Based on the data one can identify the parameters of a nonlinear model of the form

$$y(t+1) = f(y(t-n+1), \ldots, y(t), u(t)) \tag{61}$$

by for example training a quadratic neural network, where $y \in \mathbb{R}^p$, $u \in \mathbb{R}^m$, and $n \geq 1$. The value of $n-1$ is the number of delays considered in the output. If $n = 1$ then the model only considers the most recent sample $y(t)$. When $n > 1$ the model will also consider past samples of y. To have an initial idea of how many samples should be considered in the past to identify the autoregressive model one can look at the autocorrelation of the measured output and see how it decays as a function of the shift (delay) in the output. The delay at which the first significant decay happens should be the maximum delay used in the identification. We define the training matrices as

$$X = \begin{bmatrix} u^T(n) & y^T(1) & \ldots & y^T(n) \\ \vdots & \vdots & \vdots & \vdots \\ u^T(N-1) & y^T(N-n) & \ldots & y^T(N-1) \end{bmatrix}$$

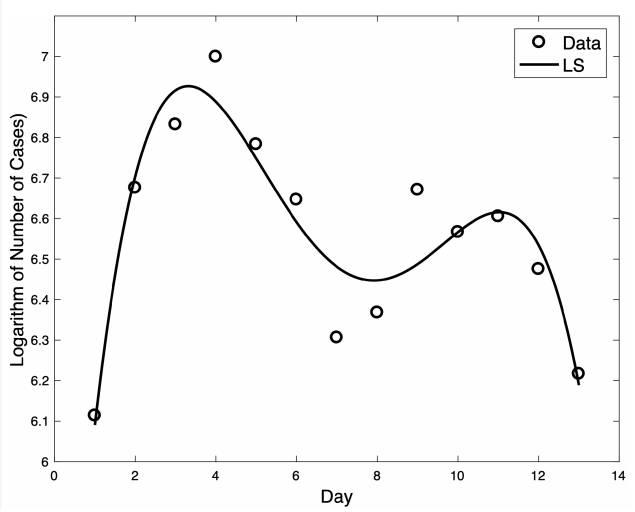

FIGURE 4. Least squares approximation of covid-19 data in the Canadian Province of Québec using a fourth order polynomial.

$$Y = \begin{bmatrix} y^T(n+1) \\ \vdots \\ y^T(N) \end{bmatrix}, \tag{62}$$

where $X \in \mathbb{R}^{(N-n)\times(m+pn)}$ and $Y \in \mathbb{R}^{(N-n)\times p}$. Each row of the matrix X is a neural network input sample and each row of the matrix Y is an output label of the training set of the quadratic neural network. It is assumed that $N \geq n+0.5(pn+m+1)(pn+m+2)$ so that the parameter estimation problem is overdetermined. It is also assumed that the collected data is rich enough in terms of persistent excitation [23], which is a typical assumption in system identification. After training the network, the input-output model is written as in equation (30) changing $\bar{x}(t)$ to $\bar{y}_u(t) = [u^T(t)\ y^T(t-n+1)\ \ldots\ y^T(t)\ 1]^T$. Defining state variables as $x_1(t) = y(t-n+1),\ \ldots,\ x_n(t) = y(t)$, a state vector $x(t) = [x_1^T(t), \ldots, x_n^T(t)]^T$, and noting that $\bar{y}_u(t) = [u^T(t)\ x^T(t)\ 1]^T = [u^T(t)\ \bar{x}^T(t)]^T$, the state space model is written as

$$x(t+1) = \begin{bmatrix} 0_{p(n-1)\times p} & I_{p(n-1)\times p(n-1)} \\ 0_{p\times p} & 0_{p\times p(n-1)} \end{bmatrix} x(t) + \begin{bmatrix} 0_{p(n-1)\times 1} \\ \mathcal{Z}(x(t), u(t)) \end{bmatrix}, \tag{63}$$

where

$$\mathcal{Z}(x(t), u(t)) = \begin{bmatrix} \bar{y}_u^T(t)\bar{Z}^1 \\ \vdots \\ \bar{y}_u^T(t)\bar{Z}^p \end{bmatrix} \bar{y}_u(t), \tag{64}$$

$$\bar{Z}^i = \begin{bmatrix} \bar{Z}^i_{uu} & \bar{Z}^i_{ux} & \bar{Z}^i_u \\ (\bar{Z}^i_{ux})^T & \bar{Z}^i_{xx} & \bar{Z}^i_x \\ (\bar{Z}^i_u)^T & (\bar{Z}^i_x)^T & \bar{Z}^i_{nn} \end{bmatrix} \tag{65}$$

for $i = 1, \ldots, p$. Expanding the quadratic forms in $\mathcal{Z}(x(t), u(t))$, the system (63)–(64) can be rewritten as

$$\bar{x}(t+1) = \bar{A}(x(t))\bar{x}(t) + \bar{B}(x(t))u(t) + E(u(t))u(t), \tag{66}$$

where $x \in \mathbb{R}^{n_x}$ with $n_x = pn$, $u \in \mathbb{R}^m$, $\bar{A}(x(t)) = \mathcal{A} + F(x(t))$, $\bar{Z}^i_{u\bar{x}} = \begin{bmatrix} \bar{Z}^i_{ux} & \bar{Z}^i_u \end{bmatrix}$, and

$$\mathcal{A} = \begin{bmatrix} A & 0_{pn\times 1} \\ 0_{1\times pn} & 1 \end{bmatrix},$$

$$A = \begin{bmatrix} 0_{p(n-1)\times p} & I_{p(n-1)\times p(n-1)} \\ 0_{p\times p} & 0_{p\times p(n-1)} \end{bmatrix},$$

$$\bar{Z}^i_{\bar{x}\bar{x}} = \begin{bmatrix} \bar{Z}^i_{xx} & \bar{Z}^i_x \\ (\bar{Z}^i_x)^T & \bar{Z}^i_{nn} \end{bmatrix},$$

$$F(x(t)) = \begin{bmatrix} 0_{p(n-1)\times(pn+1)} \\ \bar{x}^T(t)\bar{Z}^1_{\bar{x}\bar{x}} \\ \vdots \\ \bar{x}^T(t)\bar{Z}^p_{\bar{x}\bar{x}} \\ 0_{1\times(pn+1)} \end{bmatrix},$$

$$E(u(t)) = \begin{bmatrix} 0_{p(n-1)\times m} \\ u^T(t)\bar{Z}^1_{uu} \\ \vdots \\ u^T(t)\bar{Z}^p_{uu} \\ 0_{1\times m} \end{bmatrix},$$

$$\bar{B}(x(t)) = \begin{bmatrix} B(x(t)) \\ 0_{1\times m} \end{bmatrix} \tag{67}$$

where

$$B(x(t)) = 2\begin{bmatrix} 0_{p(n-1)\times m} \\ \bar{x}^T(t)\left[\bar{Z}^1_{u\bar{x}}\right]^T \\ \vdots \\ \bar{x}^T(t)\left[\bar{Z}^p_{u\bar{x}}\right]^T \end{bmatrix}. \tag{68}$$

VI. REGRESSION

Regression consists of the approximation of data points by a curve that best fits the data according to a given measure of fitness. The most common measure of fitness is the sum of the squares of the errors leading to a least squares problem. The following example highlights how least squares can be applied to regression.

Example 1: Assume that one wants to fit a straight line to a data set consisting of pairs of data points (x_i, y_i), $i = 1, \ldots, N$. Assuming a linear relationship between x_i and y_i with an additive error e_i yields

$$y_i = ax_i + b + v_i = x_i a + b + e_i.$$

Collecting all y_i, $i = 1, \ldots, N$, in a vector y, and the unknowns a, b in a vector θ, we can write an equation of the form (4) where $y = [y_1 \ldots y_N]^T$, $\theta = [a\ \ b]^T$ and

$$H = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_N & 1 \end{bmatrix}. \tag{69}$$

Letting $L = I$ and

$$M = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & 1 \end{bmatrix},$$

we will formulate a regularized least squares estimation problem and write down the formula for the solution. Then we will find the solution in the limiting case of the number of data points converging to infinity. Note that

$$H^T LH = H^T H = N \begin{bmatrix} \frac{1}{N}\sum_{i=1}^N x_i^2 & \frac{1}{N}\sum_{i=1}^N x_i \\ \frac{1}{N}\sum_{i=1}^N x_i & 1 \end{bmatrix}.$$

The inverse matrix in (11) is written as

$$(H^T LH + M)^{-1} = \frac{1}{N}\begin{bmatrix} \frac{1}{2N} + \frac{1}{N}\sum_{i=1}^N x_i^2 & \frac{1}{N}\sum_{i=1}^N x_i \\ \frac{1}{N}\sum_{i=1}^N x_i & 1 + \frac{1}{N} \end{bmatrix}^{-1}.$$

The regularized least squares solution (11) for $\hat{\theta} = [a \;\; b]^T$ is

$$\begin{bmatrix} a \\ b \end{bmatrix} = (H^T LH + M)^{-1} \begin{bmatrix} \frac{1}{N}\sum_{i=1}^N x_i y_i \\ \frac{1}{N}\sum_{i=1}^N y_i \end{bmatrix}.$$

When $N \to \infty$ this solution conveges to

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \overline{x^2} & \overline{x} \\ \overline{x} & 1 \end{bmatrix}^{-1} \begin{bmatrix} \overline{xy} \\ \overline{y} \end{bmatrix} = \frac{1}{\sigma_x^2} \begin{bmatrix} \sigma_{xy}^2 \\ \overline{x^2}\,\overline{y} - (\overline{x})(\overline{xy}) \end{bmatrix}$$

where

$$\sigma_{xx}^2 = \overline{x^2} - \overline{x}^2,$$
$$\sigma_{xy}^2 = \overline{xy} - (\overline{x})(\overline{y}),$$

and an overline on top of a variable denotes the mean value.

Example 2: We now consider a least squares approximation of the natural logarithm of the number of covid-19 cases in the Canadian Province of Québec from the first to the 13th of January 2023. The data can be found in the link: https://www.donneesquebec.ca/recherche/dataset/?groups=sante

The curve fit is a fourth degree polynomial and the obtained coefficients are

$$a_0 = 4.88, \;\; a_1 = 1.58, \;\; a_2 = -4.09 \times 10^{-1},$$
$$a_3 = 4.05 \times 10^{-2}, \;\; a_4 = -1.36 \times 10^{-3}.$$

Figure 4 shows the data overplotted with the curve fit. It is clear from the figure that the general trend of 9 out of 13 data points is captured by the curve fit.

The next example uses quadratic neural networks to perform regression of a set of data points.

Example 3: We assume a single input and single output network. Since in this case the matrix $\bar{Z}$ in equation (30) has three parameters to be estimated, we collected $N = 101$ data points, which is much greater than the number of parameters. The neural network training was performed for a quadratic activation function with parameters $a = 0.0937, b = 0.5, c = 0.4688$, a regularization coefficient of $\beta = 0.001$, and a quadratic norm loss function $l(\cdot)$ to approximate the function $f(x) = x + 2 + e$, where $e \sim N(0, 1)$. The optimization problem (27) was solved in CVX [27] interfaced with Matlab R2017a[1] and using the convex optimization solver MOSEK [28]. The symmetric matrix that describes the

[1]Matlab is a trademark of The Mathworks Inc.

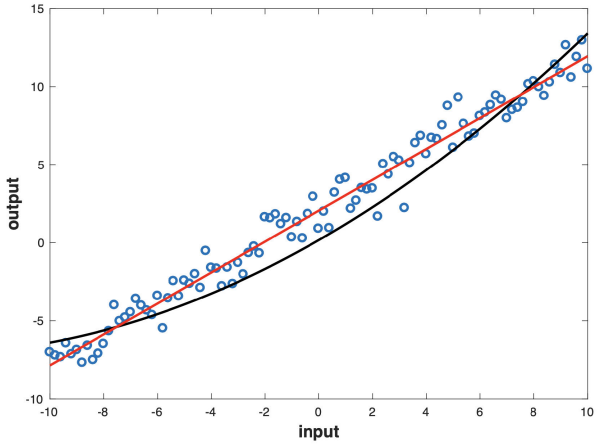

FIGURE 5. Neural network approximation of $f(x) = x + 2 + e$, where $e \sim N(0, 1)$ (black curve) and least squares approximation (red line).

neural network is

$$\bar{Z} = \begin{bmatrix} 0.0324 & 0.5241 \\ 0.5241 & 0.1619 \end{bmatrix}.$$

The neural decomposition Algorithm 2 with a tolerance of **tol**$= 10^{-5}$ leads to a neural network with two neurons and the weights

$$w_+ = \frac{1.1050}{|1.1050|} = 1, \quad w_- = \frac{-0.9357}{|-0.9357|} = -1$$

for the first layer. The weights for the second layer are

$$\alpha_+ = (1.1050)^2 = 1.221, \quad \alpha_- = -(0.9357)^2 = -0.8755.$$

The neural network output is thus given by

$$\hat{y}(x) = 1.221\sigma(x) - 0.8755\sigma(-x)$$
$$= 0.0324x^2 + 1.0482x + 0.1619.$$

The best-fitting line in the least squares sense is obtained using the method described in example 1 and is given by

$$y_{ls} = 0.9677x + 2.1302.$$

Figure 5 shows the neural network regression and its comparison to a best-fitting line in the least squares sense. It is clear from the figure that the neural network regression is not an affine line as it would be the case if one were to use standard least squares line-fitting, as seen by the red line in figure 5.

VII. CLASSIFICATION

Classification divides data into different pre-defined classes. The next example shows an application of quadratic neural networks to classification.

Example 4: In this example we will consider the classification of handwritten digits from the MNIST database [29]. This database has 60,000 digit samples for training and 10,000 digit samples for validation. Each digit image has $28 \times 28 = 784$ pixels. The first step before the classification

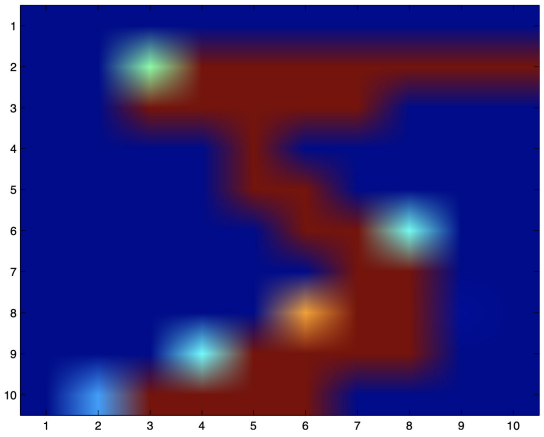

FIGURE 6. Number 5 after being downsampled.

of the digits is to downsample the images. We have chosen to only use the rows and columns 5 to 24 of each image and then downsample this smaller image so that it would only have $10 \times 10 = 100$ pixels. This downsampled image has therefore only 12.76% of the pixels of the original image. Figure 6 shows one of the downsampled images for the number 5. The reason to perform the downsampling operation is because the neural network parameters will be the entries of the $(n+1) \times (n+1)$ symmetric matrix in expression (30), where n is the number of pixels. The larger the number of pixels the larger this matrix will be and therefore the slower will be the training of the neural network. The training was done for a quadratic activation function with parameters $a = 0.0937, b = 0.5, c = 0.4688$ using a regularization coefficient of $\beta = 590$ and an infinity norm loss function $l(\cdot)$. The results of the neural network training and digit classification (testing) for different batches of 10×10 images are presented in table 1. The percentages of testing are calculated based on the number of correctly classified digits out of the $10,000$ validation set that was not used to train the neural network. The training time was measured on a MacBookAir 1.6GHz with 16GB memory. Note that although a success rate of 93.31% is smaller than the typical $96-98\%$ of a feedforward (non-convolutional) neural network using deep learning [30], only 5.37% of the total number of training images were used to achieve a 93.31% success rate. This shows that for this particular case a quadratic feedforward neural network is promising in terms of obtaining very good results with only a fraction of the total data available for training.

TABLE 1. Digit classification results.

Training Batch Size	Training Time (hours)	Success Rate (%)
100	0.21	70.89
250	0.32	82.22
500	0.44	86.54
1000	1.50	89.75
1400	2.50	90.56
1600	3.40	91.00
1920	4.30	91.53
3000	8.50	92.80
3225	19.3	93.31

VIII. IDENTIFICATION OF DYNAMIC MODELS

We start this section by going back to the example 2 and identifying the parameters of a state space model.

Example 5: We use again the data from example 2 for the number of covid-19 cases in the Canadian Province of Québec. The data runs from the 19th of December 2022 to the 13th of January 2023. With this data we will now build a (SIR) state space model with three classes: susceptible, infected, and recovered people. We assume that once a person is infected it stays infected for two weeks. After these two weeks the person is recovered and will not be susceptible again for an entire month due to immunity to infection. The number of recovered people for a given day will thus be the total number of cases two weeks prior to that day. Once a person is recovered it will not go back to being susceptible for the period of one month for which the model is being built. The model (38) with $B = 0$ will therefore have a state $x(t) = [S(t) \quad I(t)]^T$, where $S(t)$ is the number of susceptible and $I(t)$ is the number of infected people on day t. Given that the available data does not provide the number of susceptible, this number can be estimated as the total population minus the number of infected people minus the number of recovered people. The total population of Quebec is considered to be 8.5 million people for this model. Using the data and equation (48) we get the estimated A matrix to be

$$A = \begin{bmatrix} 1.00 & -0.16 \\ 0.00 & 0.89 \end{bmatrix}.$$

The entry in row two and column two is usually called the R_0 factor, which in this case is $R_0 = 0.89$. This factor means that from each nine infected people in a given day there will be $9 \times R_0 \simeq 8$ infected people in the next day.

We will now present an example on identifying the parameters of a quadratic neural network that approximates the nonlinear function of a dynamic model.

Example 6: This example considers the problem of system identification of a flexible robot arm whose input-output data with reference $[96-009]$ is available at the website of the Database for the Identification of Systems (DaISy) at the university of KU Leuven [31]. The input is the measured reaction torque of the arm structure on the ground and the output is the acceleration of the flexible arm. The applied input was a periodic sine sweep for which an output consisting of $N = 1024$ data points was collected. The input and output matrices for the neural network training are the ones from (62) with $n = 1$. The training was done for a quadratic activation function with parameters $a = 0.0937, b = 0.5, c = 0.4688$, using a regularization coefficient of $\beta = 0.01$ and an infinity norm loss function $l(\cdot)$. Only the first 122 data points were used to train the neural network, which represents approximately 12% of all

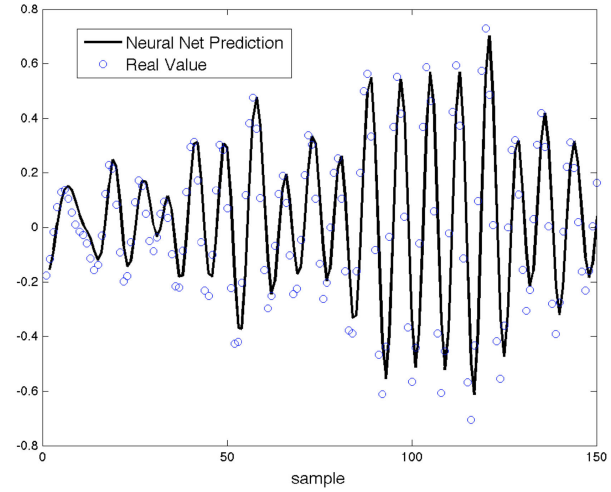

FIGURE 7. System identification (black) and data points (blue circles).

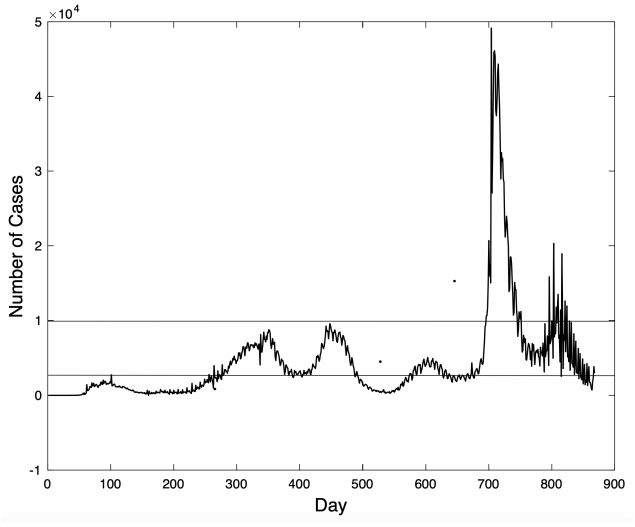

FIGURE 8. Clustering of covid-19 data for Canada over the period 26 – 01 – 2020 to 11 – 06 – 2022.

data points. The resulting (one state $x(t) = y(t)$) model is described by

$$y(t+1) = \begin{bmatrix} u(t) \\ y(t) \\ 1 \end{bmatrix}^T \bar{Z} \begin{bmatrix} u(t) \\ y(t) \\ 1 \end{bmatrix},$$

where

$$\bar{Z} = \begin{bmatrix} -0.0989 & 0.5030 & -0.1682 \\ 0.5030 & 0.1599 & 1.8265 \\ -0.1682 & 1.8265 & 0.0610 \end{bmatrix}.$$

The prediction is compared with the data in figure 7.

IX. CLUSTERING

We now use $k-$means clustering applied to the number of covid-19 cases in Canada to cluster the data into three classes: small, medium, and large number of cases. The source of the data can be found at the World Health Organization Website: https://covid19.who.int/WHO-COVID-19-global-data.csv. The data is organized in pairs (i, x_i) where i is the day and x_i is the number of cases for that day. The result of the clustering is shown in figure 8. The three clusters correspond to the data below the lowest horizontal line, the data between the two horizontal lines, and the data above the highest horizontal line. The values for the number of cases corresponding to the centroids of the three regions are

$$x_{c_1} = 848, \quad x_{c_2} = 4528, \quad x_{c_3} = 15273.$$

X. CONCLUSION

This paper provided a general framework for predictive analytics based on optimization. Four main optimization problems were introduced: least squares, the training of neural networks and support vector machines, and k-means clustering. Case studies were shown in the main applications of predictive analytics: (i) regression, (ii) time series and state space models, (iii) classification and clustering. The paper stressed that while convex optimization problems can be solved efficiently and are therefore a preferred modelling choice, the formulation of certain predictive analytics problems as convex optimization programs may be challenging. Such is the case of k-means clustering, which was disussed in section II-B. Except for this case all other problems in the paper were formulated as convex optimization programs. There are also some convex relaxations of k-means clustering in the literature. However, it is important to note that approximating a non-convex optimization as a convex problem may result in suboptimal solutions to the original problem. Such approximations should thus be considered on a case-by-case basis taking into account the benefits and drawbacks of convex relaxations. The paper also presented several examples from many different areas, which showed how predictive analytics can provide useful information to decision makers in several fields of knowledge.

REFERENCES

[1] V. Kumar and M. L. Garg, "Predictive analytics: A review of trends and techniques," *Int. J. Comput. Appl.*, vol. 182, no. 1, pp. 31–37, Jul. 2018. [Online]. Available: http://www.ijcaonline.org/archives/volume182/number1/29726-2018917434

[2] J. Dev, "Predictive analytics," *XRDS, Crossroads, ACM Mag. Students*, vol. 27, no. 3, p. 58, 2021.

[3] L. He, R. A. Levine, A. J. Bohonak, J. Fan, and J. Stronach, "Predictive analytics machinery for STEM student success studies," *Appl. Artif. Intell.*, vol. 32, no. 4, pp. 361–387, Apr. 2018.

[4] D. A. Shafiq, M. Marjani, R. A. A. Habeeb, and D. Asirvatham, "Student retention using educational data mining and predictive analytics: A systematic literature review," *IEEE Access*, vol. 10, pp. 72480–72503, 2022.

[5] N. Divyashree and K. S. P. Nandini, "Improved clinical diagnosis using predictive analytics," *IEEE Access*, vol. 10, pp. 75158–75175, 2022.

[6] N. Y. Philip, M. Razaak, J. Chang, M. Suchetha, M. O'Kane, and B. K. Pierscionek, "A data analytics suite for exploratory predictive, and visual analysis of type 2 diabetes," *IEEE Access*, vol. 10, pp. 13460–13471, 2022.

[7] R. Burbidge, M. Trotter, B. Buxton, and S. Holden, "Drug design by machine learning: Support vector machines for pharmaceutical data analysis," *Comput. Chem.*, vol. 26, no. 1, pp. 5–14, Dec. 2001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0097848501000948

[8] L. Qiu, S. Gorantla, V. Rajan, and B. C. Y. Tan, "Multi-disease predictive analytics: A clinical knowledge aware approach," *ACM Trans. Manage. Inf. Syst.*, vol. 12, no. 3, pp. 19:1–19:34, 2021.
[9] S. Ayesha, M. K. Hanif, and R. Talib, "Performance enhancement of predictive analytics for health informatics using dimensionality reduction techniques and fusion frameworks," *IEEE Access*, vol. 10, pp. 753–769, 2022.
[10] E. T. Bradlow, M. Gangwar, P. Kopalle, and S. Voleti, "The role of big data and predictive analytics in retailing," *J. Retailing*, vol. 93, no. 1, pp. 79–95, Mar. 2017.
[11] D. Broby, "The use of predictive analytics in finance," *J. Finance Data Sci.*, vol. 8, pp. 145–161, Nov. 2022.
[12] D. Kim, S. S. Roy, T. Länsivaara, R. Deo, and P. Samui, Eds., *Handbook of Research on Predictive Modeling and Optimization Methods in Science and Engineering*. Hershey, PA, USA: IGI Global, 2018.
[13] V. Kumar and M. Ram, Eds., *Predictive Analytics: Modeling and Optimization*. Boca Raton, FL, USA: CRC Press, 2021.
[14] A. Candelieri and F. Archetti, "Global optimization in machine learning: The design of a predictive analytics application," *Soft Comput.*, vol. 23, no. 9, pp. 2969–2977, May 2019.
[15] N. Krishnadoss and R. L. Kumar, "Optimizing predictive analytics model (PAM) using hyper-parameter tuning (HPT): A review," in *Proc. 5th Int. Conf. Smart Syst. Inventive Technol. (ICSSIT)*, Tirunelveli, India, Jan. 2023, pp. 1560–1563.
[16] R. D. Cook and L. Forzani, "Big data and partial least-squares prediction," *Can. J. Statist.*, vol. 46, no. 1, pp. 62–78, Mar. 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cjs.11316
[17] B. D. Liengaard, P. N. Sharma, G. T. M. Hult, M. B. Jensen, M. Sarstedt, J. F. Hair, and C. M. Ringle, "Prediction: Coveted, yet forsaken? Introducing a cross-validated predictive ability test in partial least squares path modeling," *Decis. Sci.*, vol. 52, no. 2, pp. 362–392, Apr. 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/deci.12445
[18] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.* New York, NY, USA: Association for Computing Machinery, Aug. 2017, pp. 355–364, doi: 10.1145/3077136.3080777.
[19] D. Ifenthaler and C. Widanapathirana, "Development and validation of a learning analytics framework: Two case studies using support vector machines," *Technol., Knowl. Learn.*, vol. 19, nos. 1–2, pp. 221–240, Jul. 2014, doi: 10.1007/s10758-014-9226-4.
[20] M. Ahmed, R. Seraj, and S. M. S. Islam, "The k-means algorithm: A comprehensive survey and performance evaluation," *Electronics*, vol. 9, no. 8, p. 1295, Aug. 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/8/1295
[21] A. Rakhlin and A. Caponnetto, "Stability of k-means clustering," in *Advances in Neural Information Processing Systems*, vol. 19. Cambridge, MA, USA: MIT Press, 2006. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2006/file/58191d2a914c6dae66371c9dcdc91b41-Paper.pdf
[22] H.-H. Bock, "Clustering methods: A history of k-means algorithms," in *Selected Contributions in Data Analysis and Classification*. Berlin, Germany: Springer, 2007, pp. 161–172, doi: 10.1007/978-3-540-73560-1_15.
[23] L. Ljung, *System Identification: Theory for the User*. Upper Saddle River, NJ, USA: Prentice-Hall, 1986.
[24] P. Overschee and B. Moor, *Subspace Identification for Linear Systems: Theory—Implementation—Applications*. New York, NY, USA: Springer, 1996.
[25] B. Bartan and M. Pilanci, "Neural spectrahedra and semidefinite lifts: Global convex optimization of polynomial activation neural networks in fully polynomial-time," 2021, *arXiv:2101.02429*.
[26] L. Rodrigues, "Least squares solution for training of two-layer quadratic neural networks with applications to system identification," in *Proc. 10th Int. Conf. Control, Decis., Inf. Technol.*, Valletta, Malta, Jul. 2024.
[27] M. C. Grant and S. Boyd, "The CVX user's guide release 2," CVX Res., Austin, TX, USA, Tech. Rep., Dec. 2017.
[28] *MOSEK Modeling Cookbook Release 3.2.3*, MOSEK ApS, Copenhagen, Denmark, Nov. 2021.
[29] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.
[30] P. Pauli, A. Koch, J. Berberich, P. Kohler, and F. Allgöwer, "Training robust neural networks using Lipschitz bounds," *IEEE Control Syst. Lett.*, vol. 6, pp. 121–126, 2022.
[31] B. D. Moor, P. D. Gersem, B. D. Schutter, and W. Favoreel, "Daisy: A database for identification of systems," *J. A*, vol. 38, no. 3, pp. 4–5, 1997.

LUIS RODRIGUES (Senior Member, IEEE) received the Licenciatura and M.Sc. degrees in electrical and computer engineering from IST, Technical University of Lisbon, and the Ph.D. degree in aeronautics and astronautics from Stanford University, in 2002.

He was a Consultant in speech modeling and recognition with Eliza Corporation, USA, and a Flight Simulation Project Manager with Ydreams, Portugal. He is currently a Professor with the Department of Electrical and Computer Engineering, Concordia University, and a member of the Concordia Institute of Aerospace Design and Innovation. His research interests include optimization, control, and machine learning with applications to aerospace, robotics, and autonomous vehicles.

SIDNEY N. GIVIGI (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Carleton University, Ottawa, ON, Canada.

He was an Assistant Professor and an Associate Professor with the Department of Electrical and Computer Engineering, Royal Military College of Canada, Kingston, ON, Canada, from 2009 to 2014 and from 2014 to 2019, respectively. Since 2019, he has been an Associate Professor with the School of Computing, Queen's University, Kingston. His research interests include control systems, robotics, autonomous vehicles, and machine learning.