Least-Squares Linear Dilation-Erosion Regressor
Trained using Stochastic Descent Gradient or
the Difference of Convex Methods

Angelica Lourenço Oliveira\textsuperscript{1,2} and
Marcos Eduardo Valle\textsuperscript{3,4}

University of Campinas, Campinas-SP, Brazil.
ra211686@ime.unicamp.br and valle@ime.unicamp.br

Abstract. This paper presents a hybrid morphological neural network for regression tasks called linear dilation-erosion regression (\(\ell\)-DER). In few words, an \(\ell\)-DER model is given by a convex combination of the composition of linear and elementary morphological operators. As a result, they yield continuous piecewise linear functions and, thus, are universal approximators. Apart from introducing the \(\ell\)-DER models, we present three approaches for training these models: one based on stochastic descent gradient and two based on the difference of convex programming problems. Finally, we evaluate the performance of the \(\ell\)-DER model using 13 regression tasks. Although the approach based on SDG revealed faster than the other two, the \(\ell\)-DER trained using a disciplined convex-concave programming problem outperformed the others in terms of the least mean absolute error score.

Keywords: Morphological neural network · continuous piecewise linear function · regression · DC optimization.

1 Introduction

Dilations and erosions are the elementary operations of mathematical morphology, a non-linear theory widely used for image processing and analysis\textsuperscript{13,25}. In the middle 1990s, Ritter et al. proposed the first morphological neural networks whose processing units, the neurons, perform dilations and erosions\textsuperscript{21,22}. In general terms, morphological neurons are obtained by replacing the usual dot product with either the maximum of sums or the minimum of sums. Because of the maximum and minimum operations, morphological neural networks are usually cheaper than traditional models. However, training morphological neural networks are often a big challenge because of the non-differentiability of the extreme operations\textsuperscript{29}. This paper addresses this issue by investigating different methods for training a hybrid morphological neural network for regression tasks. Precisely, we focus on training algorithms for the so-called linear dilation-erosion perceptron.

\textsuperscript{*} This work was supported in part by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.
A dilation-erosion perceptron (DEP) is a hybrid morphological neural network obtained by a convex combination of dilations and erosions [1]. Despite its application in regression tasks such as time-series prediction and software development cost estimation [1, 3], the DEP model has an inherent drawback: As an increasing operator, it implicitly assumes an ordering relationship between inputs and outputs [30]. Fortunately, one can circumvent this problem by adding neurons that perform anti-dilations and anti-erosions [27]. For example, considering the importance of dendritic structures, Ritter and Urcid presented a single morphological neuron that circumvents the limitations of the DEP model [23].

Alternatively, Valle recently proposed the reduced dilation-erosion perceptron (r-DEP) using concepts from multi-valued mathematical morphology [30]. In few words, an r-DEP is obtained by composing an appropriate transformation with the DEP model, i.e., the inputs are transformed before they are fed to the DEP model. However, choosing the proper transformation is the most challenging task to design an efficient r-DEP model. As a solution, Oliveira and Valle recently proposed the so-called linear dilation-erosion perceptron (ℓ-DEP) by considering linear mappings instead of arbitrary transformations [18]. Interestingly, the linear dilation-erosion perceptron is equivalent to a maxout network investigated by Goodfellow et al [10]. Also, the ℓ-DEP is closely related to one of two hybrid morphological neural networks investigated by Hernández et al. for big data classification [14]. From a mathematical point of view, the ℓ-DEP yields a continuous piecewise linear function. Thus, like many traditional neural networks, they are universal approximators; that is, an ℓ-DEP model can approximate a continuous function within any desired accuracy in a compact region in a Euclidean space [31].

As pointed out in the previous paragraph, the ℓ-DEP is equivalent to the maxout network. Like traditional neural networks, maxout networks are usually trained using the stochastic gradient descent (SGD) method [9, 10]. Henández et al. also used the SDG method for training their hybrid morphological neural networks [14]. In contrast to the works mentioned above, Ho et al. formulated the learning of a continuous piecewise linear function as a difference of convex (DC) programming problem [15, 16]. This paper revises the difference of convex algorithm (DCA) applied for training the ℓ-DEP model for regression tasks. Apart from the SDG and DCA-based learning rules, we also formulate the training of an ℓ-DEP model as a disciplined convex-concave procedure (DCCP) [24]. We evaluate and compare the performance of the resulting ℓ-DEP model using several regression tasks.

The paper is organized as follows. The following section reviews some basic concepts regarding DC optimization, including definitions and properties of convex functions and DC functions. Section 3 presents the ℓ-DER model, while three different approaches for training this model are addressed in Section 4. Computational experiments comparing the performance of the ℓ-DER model trained using the three approaches are given in Section 5. The paper finishes with some remarks in Section 6.
2 Basic Concepts on DC Optimization

DC optimization aims to optimize the difference of two convex functions, a broad class of non-convex functions that enjoy interesting and useful properties \cite{12, 29}. As remarked by Shen et al. \cite{24}, applications of DC programs include signal processing, machine learning, computer vision, and statistics. The following presents the concepts of DC functions. We subsequently address the two kinds of DC optimization problems considered in this paper.

**Definition 1 (DC function).** Let $f : \mathbb{C} \rightarrow \mathbb{R}$ be a real-valued function defined in a convex set $\mathbb{C}$. We say that $f$ is a DC function in $\mathbb{C}$ if there exist convex functions $g, h : \mathbb{C} \rightarrow \mathbb{R}$ such that $f$ can be expressed as

$$f(\alpha) = g(\alpha) - h(\alpha), \quad \forall \alpha \in \mathbb{C}. \quad (1)$$

Writing $f := g - h$ is called the DC decomposition of $f$ and the functions $g$ and $h$ are referred to as the DC components of $f$.

DC optimization problems are problems in which the objective and constraints are described by DC functions. In this paper, we focus on the following two DC optimization problems: Unconstrained DC optimization problem formulated as

$$\min_{\alpha \in \mathbb{R}^n} f(\alpha) = g(\alpha) - h(\alpha), \quad (2)$$

and constrained DC optimization problem given by

$$\min_{\alpha \in \mathbb{R}^n} f(\alpha) = g(\alpha) - h(\alpha) \quad \text{s.t.} \quad F(\alpha) = G(\alpha) - H(\alpha) \leq 0, \quad (3)$$

where $g : \mathbb{R}^n \rightarrow \mathbb{R}$, $h : \mathbb{R}^n \rightarrow \mathbb{R}$, $G : \mathbb{R}^n \rightarrow \mathbb{R}^m$, and $H : \mathbb{R}^n \rightarrow \mathbb{R}^m$ are all convex functions. In this paper, we consider two methods for solving DC optimization problems. The first, which solves the unconstrained DC problem (2), is the difference of convex optimization algorithm (DCA). The second, which solves the constrained problem (3), is called disciplined convex-concave program (DCCP). In general terms, DC optimization methods take advantage of the convexity of the DC components $g$ and $h$ of $f$. Thus, before addressing these methods, let us review some important properties of convex functions.

2.1 Some Properties of Convex Functions

We begin by reviewing the concepts of subgradient and sub-differentiability. Subgradients, which generalize the notion of gradients, are well-defined even for non-smooth convex functions.

**Definition 2 (Subgradient and Subdifferential).** Let $f : \mathbb{C} \subset \mathbb{R}^n \rightarrow \mathbb{R}$ be a convex function. A subgradient of the $f$ at $\alpha \in \mathbb{C}$ is a vector $\beta \in \mathbb{R}^n$ such that

$$f(z) \geq f(\alpha) + \langle \beta, z - \alpha \rangle, \quad \forall z \in \mathbb{C}^n. \quad (4)$$

The set of all subgradients of $f$ at $\alpha$, denoted by $\partial f(\alpha)$, is called the subdifferential of $f$ in $\alpha$. Formally, the subdifferential of $f$ at $\alpha$ is

$$\partial f(\alpha) := \{ \beta \in \mathbb{R}^n : f(z) \geq f(\alpha) + \langle \beta, z - \alpha \rangle, \quad \forall z \in \mathbb{R}^n \}. \quad (5)$$
From (4), for any $\beta \in \partial f(\alpha)$, the affine function defined by $\ell(z) := f(\alpha) + \langle \beta, z - \alpha \rangle$ for all $z \in \mathbb{R}^n$ is a lower approximation of $f$ at $\alpha$. Geometrically, $\ell$ determines a hyperplane tangent to the convex function $f$ in $\alpha$. Moreover, then the graph of $f$ is always above the graph of $\ell$, i.e., the tangent plane.

Finding subgradients of a convex function may not be an easy task. Hopefully, some results facilitate the determination of subgradients. For example, Frechel-Young inequality yields an efficient way to find subgradients of a broad class of convex functions. To present the Frechel-Young inequality, we need a few more concepts.

**Definition 3 (Proper Function).** Let $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ be a convex function. We say that $f$ is a proper function if $f$ assumes at least a finite value and does not assume any value equal to $-\infty$. In other words, $f(\alpha) < +\infty$ for some $\alpha \in \mathcal{C}$ and $f(\alpha) > -\infty$ for all $\alpha \in \mathcal{C}$.

**Definition 4 (Conjugate Function).** Given a function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$, its conjugate is the function $f^* : \mathbb{R}^n \to \bar{\mathbb{R}}$ defined by

$$f^*(\beta) = \sup_{\alpha \in \mathbb{R}^n} \{ \langle \alpha, \beta \rangle - f(\alpha) \}, \forall \beta \in \mathbb{R}^n. \quad (6)$$

**Definition 5 (Lower Semicontinuous Functions).** A convex function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ is lower semicontinuous (lsc) function if $f^{**} = f$.

The space of all proper and lower semicontinuous functions on $\mathcal{C}$ is denoted by $\Gamma_0(\mathcal{C})$.

**Proposition 1. (Fenchel-Young Inequality)** If $f \in \Gamma_0(\mathcal{C})$, that is, $f$ is a proper lsc function, then

$$h(\alpha) + h^*(\beta) \geq \langle \alpha, \beta \rangle, \quad (7)$$

any $\beta \in \mathbb{R}^n$ and $\alpha \in \mathcal{C}$. Moreover, the equality holds if and only if $\beta \in \partial f(\alpha)$.

The Fenchel-Young equality result is known as the conjugate subgradient theorem. The conjugate subgradient theorem is used to find a subgradient of a function $f \in \Gamma_0(\mathcal{C})$.

### 2.2 Difference of Convex Optimization Algorithm

In [28], Pham Dinh Tao introduced DCA by extending the gradient algorithm used for convex maximization to DC programming. Since 1994, with the joint work of Le Thi Hoai An and Pham Dinh Tao, the DCA algorithm has been developed and improved, both in theoretical and computational aspects. In general terms, the DCA is a subgradient optimization method used to solve unconstrained DC optimization problems. Furthermore, it is based on local optimization and dual DC programming.
Algorithm 1: DCA

Input: Convex functions: $g, f$
Output: $\alpha^*, \beta^*$ (Primal and Dual solution)
Initialize: $\alpha_0 \in \mathbb{R}^n$

$k = 0$
repeat
  Compute $\beta_k \in \partial h(\alpha_k)$
  Compute $\alpha_{k+1} \in \partial g^*(\beta_k)$
  $k = k + 1$
until converge;
return $\alpha^* = x_k$ and $\beta^* = \beta_{k-1}$

Consider a proper unconstrained DC problem given by (2) that can also be alternatively written as
\[
\inf_{\alpha \in \mathbb{R}^n} \{ f(\alpha) = g(\alpha) - h(\alpha) \},
\]where $g, h \in \Gamma_0(\mathbb{R}^n)$. We would like to point out that (8) is the standard form of a primal DC programming problem. The standard form of the dual DC programming problem for (8) is given by the following DC problem
\[
\inf_{\beta \in \mathbb{R}^n} \{ f^*(\beta) = h^*(\beta) - g^*(\beta) \},
\]Let $\lambda$ and $\lambda^*$ the values of the solutions of the problems (9) and (8). It is easy to show that $\lambda = \lambda^*$. In other words, solving problem (8) is equivalent to solving problem (9).

The DCA algorithm can be used to solve both the primal and dual problems. In order to introduce the DCA algorithm, let $\alpha_k$ be a convergent sequence and its $\alpha^*$ its limit, that is, $\alpha_k \rightarrow \alpha^*$. Since $h \in \Gamma_0(\mathbb{R}^n)$ and exists a bounded sequence $\beta_k \in \partial h(\alpha_k) \neq \emptyset$, then $\lim_{k \rightarrow \infty} h(\alpha_k) = h(\alpha^*)$. This result is the key concept for the formulation of the DCA method described by Algorithm 1. In general terms, the Algorithm 1 seeks to find two convergent sequences $\{\alpha_k\}$ and $\{\beta_k\}$ such that their accumulation points are approximations for the local solutions of the primal and dual problems, respectively. In other words, the algorithm yields two sequences $\{\alpha_k\}$ and $\{\beta_k\}$ such that $\{f_k = f(\alpha_k)\}$ and $\{f^*_k = f^*(\beta_k)\}$ are both decreasing sequences.

One of the challenges of in Algorithm 1 is to find the subgradients $\beta_k \in \partial h(\alpha_k)$ and $\alpha_{k+1} \in \partial g^*(\beta_k)$ in each iteration. Since solving the primal problem is equivalent to solving the dual problem, the choice of $\beta_k$ will be arbitrary and the choice of $\alpha_{k+1}$ will be given by solving an optimization problem. Since $g^{**} = g \in \Gamma_0(\mathbb{C})$, $\mathcal{C} \subset \mathbb{R}^n$, from the conjugate subgradient theorem the following equivalences hold for any $k \in \mathbb{N}$:
\[
\alpha_{k+1} \in \partial g^*(\beta_k) \iff g^*(\beta_k) + g(\alpha_{k+1}) = \langle \alpha_{k+1}, \beta_k \rangle \\
\iff \beta_k \in \partial g(\alpha_{k+1}) \\
\iff g(\alpha_{k+1}) - \langle \beta_k, \alpha_{k+1} \rangle \leq g(\alpha) - \langle \beta_k, \alpha \rangle, \forall \alpha \in \mathcal{C}.
\]
Furthermore, the following identities hold:

\[
\min_{\alpha \in C} \{g(\alpha) - \langle \beta_k, \alpha \rangle\} = \min_{\alpha \in C} \{g(\alpha) - \langle \beta_k, \alpha \rangle - h(\alpha_k) + \langle \beta_k, \alpha_k \rangle\} = \min_{\alpha \in C} \{g(\alpha) - [h(\alpha_k) + \langle \beta_k, \alpha - \alpha_k \rangle]\}.
\]

Therefore, the sequence \(\alpha_{k+1}\) can be found by solving the convex optimization problem given by

\[
\alpha_{k+1} = \arg \min_{\alpha \in C} \{g(\alpha) - [h(\alpha_k) + \langle \beta_k, \alpha - \alpha_k \rangle]\}.
\]

(10)

The limit of the sequence \(\{\alpha_k\}\) is an approximation to the solution of the primal problem. The convergence analysis and other properties of Algorithm 1 can be found in [8,28].

2.3 Disciplined Convex-Concave Programming

Disciplined convex-concave programming (DCCP) refers to a methodology introduced by Shen et al. for solving the difference of convex problems [24]. In few words, DCCP combines concave-convex programming (CCP) [32] with disciplined convex programming (DCP), allowing the latter to deal with DC optimization problems. Let us briefly address some fundamental characteristics of the DCCP methodology. We begin by reviewing concave-convex programming.

Concave-convex programming is a majorization-minimization methodology that uses convex optimization tools to find local optimum for DC problems through of a sequence of convex subproblems [11, 32]. The CCP methods can solve the non-convex problem whenever the objective and constraints are DC functions.

Consider the constrained DC problem given by (3). Roughly speaking, a CCP method approximates the concave terms \(-h\) in the objective and \(-H\) in the constraints by convex majorant functions. Then, the resulting subproblem is a convex optimization problem that can be solved using convex optimization techniques.

In [17] the authors introduced a variation of the CCP called penalty CCP, in which the convex majorant that approximates the concave terms are affine functions, that is, the functions \(-h\) and \(-H\) are approximated from below by affine functions. The penalty CCP method is presented described in Algorithm [2] where the vector-valued functions \(G, H : \mathbb{R}^n \to \mathbb{R}^m\) are written as \(G(\alpha) = (g_1(\alpha), ..., g_m(\alpha))\) and \(H(\alpha) = (h_1(\alpha), ..., h_m(\alpha))\). In Algorithm [2], the concave terms are explicitly linearized using subgradients. Moreover, to simplify the notation, the objective function is decomposed by \(f = g_0 - h_0\), i.e., \(g_0 = g\) and \(h_0 = h\).

As pointed out previously, DCCP is obtained including CCP in the disciplined convex programming (DCP) methodology [11]. Briefly, DCP is a system of pre-established rules for constructing mathematical expressions with known curvatures widely used by convex optimization libraries such as CVX, CVXPY, and
Algorithm 2: PENALTY CCP

**Input:** Convex functions: \( g_0, \ldots, g_m \) and \( f_0, \ldots, f_m \).

**Output:** \( \alpha^* \) (Solution)

**Initialize:** \( \alpha_0 \in \mathbb{R}^n, \mu > 1, t_0 > 0, t_{max} > 0, \) and \( k = 0 \).

repeat

1. Compute \( \beta_i \in \partial h_i(\alpha_k) \), for all \( i = 0, \ldots, m \).
2. Solve the convex problem:

\[
\begin{aligned}
\alpha_k &= \arg\min_{\alpha} g_0(\alpha) - h_0(\alpha_k) - \langle \beta_0, \alpha - \alpha_k \rangle + t_k \sum_{i=1}^{m} s_i \\
&\text{s.t. } g_i(\alpha) \leq h_i(\alpha_k) + \langle \beta_i, \alpha - \alpha_k \rangle + s_i, \quad \forall i = 1, \ldots, m, \\
s_i &\geq 0, \quad \forall i = 1, \ldots, m.
\end{aligned}
\]

3. \( t_{k+1} = \min\{\mu t_k, t_{max}\} \)
4. \( k = k + 1 \)

until converge;

return \( \alpha^* = \alpha_k \)

Convex.jl [7]. In mathematical terms, DCP deals with optimization problems of the following kind

\[
\minimize_{\alpha \in \mathbb{R}^n} f(\alpha) \quad \text{s.t. } g_i(\alpha) \sim h_i(\alpha), \quad \forall i = 1, \ldots, m, \tag{11}
\]

where the curvature of the functions \( f, g_1, \ldots, g_m, h_1, \ldots, h_m \) are known and the following statements must hold:

1. If (11) is a minimization problem, then \( f \) must be convex.
2. If (11) is a maximization problem, then \( f \) must be concave.
3. If \( \sim \) equals \( \leq \), \( g_i \) must be convex and \( h_i \) must be concave for all \( i = 1, \ldots, m \).
4. If \( \sim \) equals \( \geq \), \( g_i \) must be concave and \( h_i \) must be convex for all \( i = 1, \ldots, m \).
5. If \( \sim \) equals \( = \), then \( g_i \) and \( h_i \) must both be affine for all \( i = 1, \ldots, m \).

The reference [11] provides further details on disciplined convex programming.

Finally, a disciplined concave-convex programming (DCCP) problem is similar to (11) but with some relaxed versions of the five statements listed above, which allows generalizing the use of DCP for DC problems. Like in DCP, the curvatures of the objective function and the constraints functions are previously known in a DCCP problem.

Knowing the curvatures, DCCP can cope with non-convex problems as long as the functions are written as DC functions in a disciplined way. Then, the Algorithm 2 is used to solve the DC problem.

3 Linear Dilation-Erosion Regressor

Predictive classification models are used to categorize information based on a set of historical data. Predictive regression models are used to solve curve-fitting
problems whose goal is to find a function that best fits a specific curve for a given set of data. This adjustment can be helpful in forecasts or estimates outside the data set.

Recently, [18] introduced the linear dilation-erosion perceptron (ℓ-DEP) for classification tasks. A linear dilation-erosion perceptron is given by a convex combination of linear transformations and two elementary operators from mathematical morphology [1,21]. Let us review the main concepts from mathematical morphology and the ℓ-DEP model. We will subsequently present the linear dilation-erosion regressor, the predictive model of the regression type corresponding to the ℓ-DEP classifier.

Mathematical morphology is mainly concerned with non-linear operators defined on complete lattices [13,25]. Complete lattices are partially ordered sets with well-defined supremum and infimum operations [4]. Dilations and erosions are the elementary operators from mathematical morphology. Given complete lattices $L$ and $M$, a dilation $\delta : L \rightarrow M$ and an erosion $\varepsilon : L \rightarrow M$ are operators such that $\delta(\sup X) = \sup \{\delta(x) : x \in X\}$ and $\varepsilon(\inf X) = \inf \{\varepsilon(x) : x \in X\}$ for all $X \in L$ [13]. For example, given vectors $a, b \in \mathbb{R}^n$, the operators $\delta_a, \varepsilon_b : \bar{\mathbb{R}}^n \rightarrow \bar{\mathbb{R}}$ given by

$$\delta_a(x) = \max_{j=1:n} \{a_j + x_j\} \quad \text{and} \quad \varepsilon_b(x) = \min_{j=1:n} \{b_j + x_j\}, \quad (12)$$

for all $x \in \bar{\mathbb{R}}^n$ are respectively a dilation and an erosion [27].

A dilation-erosion perceptron (DEP) is given by a convex combination of a dilation and an erosion defined by (12). The reduced dilation-erosion perceptron (r-DEP) proposed recently by Valle is an improved version of the DEP model obtained using concepts from vector-valued mathematical morphology [30]. The ℓ-DEP model is a particular but powerful r-DEP classifier [18]. Formally, given a one-to-one mapping $\sigma$ from the set of binary class labels $C$ to $\{+1,-1\}$, a ℓ-DEP classifier is defined by the equation $y = \sigma^{-1} f^{\ell}(x)$, where $f : \mathbb{R} \rightarrow \{-1, +1\}$ is a threshold function and $\tau^{\ell} : \mathbb{R}^n \rightarrow \mathbb{R}$ is the decision function given by

$$\tau^{\ell}(x) = \delta_{\mathbf{a}}(Wx) - \delta_{\mathbf{b}}(Mx). \quad (13)$$

Equivalently, the decision function $\tau^{\ell}$ satisfies

$$\tau^{\ell}(x) = \max_{i=1:r_1} \{w_i^T x + a_i\} - \max_{j=1:r_2} \{m_j^T x + b_j\}, \quad (14)$$

where $\mathbf{a} = (a_1, \ldots, a_{r_1}) \in \mathbb{R}^{r_1}$ and $\mathbf{b} = (b_1, \ldots, b_{r_2}) \in \mathbb{R}^{r_2}$, and $w_i^T$ and $m_j^T$ are rows of $W \in \mathbb{R}^{r_1 \times n}$ and $M \in \mathbb{R}^{r_2 \times n}$, respectively. From the last identity, we can identify $\tau^{\ell}$ with a piece-wise linear function [31]. Moreover, from Theorem 4.3 in [10], the decision function $\tau^{\ell}$ is an universal approximator, i.e., it is able to approximate any continuous-valued function from a compact set on $\mathbb{R}^n$ to $\mathbb{R}$ [10, 26]. As a consequence, an ℓ-DEP model can theoretically solve any binary classification problem.

Because the decision function of ℓ-DEP model is a universal approximator, $\tau^{\ell}$ given by (13) can also be used as a predictive model for regression tasks. In other
words, it is possible to use the $\tau^\ell$ as the prediction function that maps a set of independent variables in $\mathbb{R}^n$ to a dependent variable in $\mathbb{R}$. In this case, we refer to $\tau^\ell : \mathbb{R}^n \to \mathbb{R}$ as a linear dilation-erosion regressor ($\ell$-DER). In this paper, the parameters $(w_i^T, a_i) \in \mathbb{R}^{n+1}$ and $(m_j^T, b_j) \in \mathbb{R}^{n+1}$, for $i = 1 : r_1$ and $j = 1 : r_2$, are determined using a training set by minimizing the squares of the difference between the predicted and desired values. The following section address three different approaches for training an $\ell$-DER model.

4 Three Approaches for Training $\ell$-DER Models

In this section we present approaches for training an $\ell$-DER model using set $\mathcal{T} = \{(x_i, y_i) : i = 1 : m\} \subset \mathbb{R}^n \times \mathbb{R}$, called the training set. Precisely, the goal is to find the parameters of an $\ell$-DER model such that the estimate $\tau^\ell(x_i)$ approaches the desired output $y_i$ according to some loss function. Recall that the parameters of an $\ell$-DEP regressors are the matrices $W \in \mathbb{R}^{r_1 \times n}$ and $M \in \mathbb{R}^{r_2 \times n}$ as well as the vectors $a = (a_1, \ldots, a_{r_1}) \in \mathbb{R}^{r_1}$ and $b = (b_1, \ldots, b_{r_2}) \in \mathbb{R}^{r_2}$. To simplify the exposition, the parameters of an $\ell$-DER are also arranged in a vector

$$\alpha = (w_1^T, a_1, \ldots, w_{r_1}^T, a_{r_1}, m_1^T, b_1, \ldots, m_{r_2}^T, b_{r_2}) \in \mathbb{R}^{(r_1+r_2)(n+1)},$$

(15)

where $w_i^T$ and $m_i^T$ are the rows of $W \in \mathbb{R}^{r_1 \times n}$ and $M \in \mathbb{R}^{r_2 \times n}$, respectively. During training, an $\ell$-DER is interpreted as a function of its parameters, that is, $\tau^\ell(x) \equiv \tau^\ell(x; \alpha)$.

In this paper, the widely used mean squared error (MSE) defined as follows using the training set $\mathcal{T} = \{(x_i, y_i) : i = 1, \ldots, m\} \subset \mathbb{R}^n \times \mathbb{R}$:

$$MSE(\mathcal{T}, \alpha) = \frac{1}{m} \sum_{i=1}^{m} (y_i - \tau^\ell(x_i; \alpha))^2,$$

(16)

is considered as the loss function. As a consequence, the parameters of the $\ell$-DER are determined by solving the optimization problem

$$\min_{\alpha} \frac{1}{m} \sum_{i=1}^{m} (y_i - \tau^\ell(x_i; \alpha))^2.$$

(17)

In the following subsections, we will present three approaches for solving (17). The first two approaches deal with unconstrained optimization problems. Precisely, the first one applies an stochastic gradient descent (SGD) algorithm directly for solving (17) [5]. The second approach, which has been proposed [16], uses DCA. The last approach, inspired by the training of the $\ell$-DEP classifier [18], uses DCCP for solving (17).

4.1 Approach Based on Stochastic Gradient Descent Method

First of all, note that the $\ell$-DER model $\tau^\ell$ given by (13) corresponds to a maxout network with two maxout units [10]. Like many modern neural networks,
we may train an $\ell$-DER using the iterative method based on the descent gradient method. Precisely, we use the stochastic gradient descent (SGD) method to minimize $\ell$. At this point, we would like to recall that the SGD method uses gradients of the objective function at a point to find a local minimum of the loss function. Despite the maximum operation in $\tau$, the set of points where the loss function (17) is not differentiable has measure zero. In other words, the loss function is differentiable almost everywhere, and the non-differentiability of the maximum operation is usually not a problem for using this approach. The reader who wants to know more about the SGD method is encouraged reading [5].

Because SGD is implemented in machine learning libraries like tensorflow and pytorch, we believe this is the most straightforward procedure among the three approaches considered in this work. Therefore, let us turn our attention to the other two approaches.

4.2 Approach Based on the Difference of Convex Algorithm

Let us begin by noting that $\tau(x)$ given by (13) is a DC function because $\delta_a(Wx)$ and $\delta_b(Mx)$ are both convex functions. In the following, we write the square of the difference $\tau(x_i; \alpha) - y_i$, as a DC function for all $i = 1, \ldots, m$.

Because there are infinite DC decompositions for a single DC function, it is possible to define a convex function $\phi_i$ such that

\[ (\tau^\ell(x_i; \alpha) - y_i)^2 = (\tau^\ell(x_i; \alpha) - y_i + \phi_i(\alpha) - \phi_i(\alpha))^2 \]
\[ = (\delta_a(Wx_i) - y_i + \phi_i(\alpha) - (\phi_i(\alpha) + \delta_b(Mx_i)))^2 \]

can be written as a DC function. From the identity $(x-y)^2 = 2(x^2+y^2) - (x+y)^2$, we conclude that

\[ (\tau^\ell(x_i; \alpha) - y_i)^2 = 2((\tau_1(x_i; \alpha) + \phi_i(\alpha))^2 + (\phi_i(\alpha) + \tau_2(x_i; \alpha))^2) \]
\[ - (\tau_1(x_i; \alpha) + \tau_2(x_i; \alpha) + 2\phi_i(\alpha))^2 \]

where $\tau_1(x_i; \alpha) = \delta_a(Wx_i) - y_i$ and $\tau_2(x_i; \alpha) = \delta_b(Mx_i)$ are convex functions. Thus, the mean squared error given by (17) admits a DC decomposition $MSE(T, \alpha) = G(\alpha) - H(\alpha)$ where

\[ G(\alpha) = \frac{2}{m} \sum_{i=1}^{m} [(\tau_1(x_i; \alpha) + \phi_i(\alpha))^2 + (\tau_2(x_i; \alpha) + \phi_i(\alpha))^2] \]
\[ H(\alpha) = \frac{1}{m} \sum_{i=1}^{m} [(\tau_1(x_i; \alpha) + \tau_2(x_i; \alpha) + 2\phi_i(\alpha))^2] \]

with

\[ \phi_i(\alpha) = \max\{\tau_1(x_i; \alpha) + \langle \beta_1, \alpha - \bar{\alpha} \rangle, \tau_2(x_i; \alpha) + \langle \beta_2, \alpha - \bar{\alpha} \rangle\} \] (18)

for an arbitrary vector $\bar{\alpha} \in \mathbb{R}^{(r_1+r_2)(n+1)}$ and $\beta_j \in \partial \tau_j(x_i; \alpha)$, $j = 1, 2$. Because $MSE(T, \alpha)$ is a DC function, DCA can be used to minimize (17).
As pointed out previously in Section 2, one of the challenges of the DCA algorithm is to find elements \( \beta_t \in \partial H(\alpha_t) \) and \( \alpha_{t+1} \in \partial G^*(\beta_t) \) in each iteration \( t \). For training the \( \ell \)-DER, in particular, these elements are determined as follows:

Let \( v_i ∈ (v_1, ..., v_m) ∈ R^{(r_1+r_2)(n+1)} \) for \( i = 1, ..., m \) and \( s = 1, ..., r_1 + r_2, \) where

\[
v_{k,s} = \begin{cases} (x_i, 1), & k = s, \\ (0, 0), & \text{otherwise}, \end{cases}
\]

for \( k = 1, ..., r_1 + r_2 \) and \( 0 = (0, ..., 0) ∈ R^n \). Also, define \( v^i = v_{i,j_1} - v_{i,j_1+j_2} \) where

\[
j_1 = \arg \max_{l=1, ..., r_1} \{ w_j^T x_i + a_j \} \quad \text{and} \quad j_2 = \arg \max_{l=1, ..., r_2} \{ m_j^T x_i + b_j \}.
\]

Inspired by [16], the choice of \( \beta_t \in \partial H(\alpha_t) \) will be given by

\[
\beta_t = 2 \sum_{i=1}^{m} (r^f(x_i; \alpha_t) - y_i) v^i ∈ R^{(r_1+r_2)(n+1)},
\]

From the properties of the conjugate functions, \( \alpha_{t+1} ∈ \partial G^*(\beta_t) \) is obtained by solving the quadratic optimization problem

\[
\begin{aligned}
\text{minimize} & \quad \|q\|_2^2 + \|p\|_2^2 - \langle \beta_t, \alpha \rangle \\
\text{s.t.} & \quad \langle v^{l,l} - v^{i,j_2}, \alpha \rangle ≤ q_l, \quad l = 1, ..., r_1, \quad i = 1, ..., m, \\
& \quad \langle v^{l,l} - v^{i,j_1}, \alpha \rangle ≤ q_l + y_i, \quad l = 1, ..., r_1, \quad i = 1, ..., m, \\
& \quad \langle v^{l,l} - v^{i,j_1+j_2}, \alpha \rangle ≤ p_l - y_i, \quad l = 1, ..., r_2, \quad i = 1, ..., m, \\
& \quad \langle v^{l,l} - v^{i,j_2}, \alpha \rangle ≤ p_l, \quad l = 1, ..., r_2, \quad i = 1, ..., m.
\end{aligned}
\]

The stopping criterion of DCA used for training the \( \ell \)-DER is \( |MSE(T, \alpha_t) - MSE(T, \alpha_{t-1})| ≤ \epsilon (1 + MSE(T, \alpha_{t-1})) \), with \( \epsilon = 10^{-6} \).

### 4.3 Approach Based on Disciplined Convex-Concave Programming

Inspired by methodology developed by Charisopoulos and Maragos for training morphological perceptrons [6], we will reformulate the unrestricted optimization problem [17] as a constrained DC problem. Precisely, let \( \xi_i = y_i - \tau^f(x_i), \) for \( k = 1, ..., m. \) Then, the unrestricted problem [17] corresponds to minimizing \( \frac{1}{m} \| \xi \|_2^2 \) subject to the constraints \( \tau^f(x_i) = y_i - \xi_i \) for all \( i = 1, ..., m. \) In other words, the \( \ell \)-DER can be trained by solving the following DCCP problem

\[
\begin{aligned}
\text{minimize} & \quad \frac{1}{m} \| \xi \|_2^2 \\
\text{s.t.} & \quad \delta_a(Wx_i) + \xi_i = \delta_b(Mx_i) + y_i, \quad i = 1, ..., m.
\end{aligned}
\]

Note that the objective function in (21) is a convex quadratic function. Moreover, the functions at both sides of the equality constraints have known concavity; they are convex functions. Therefore, the optimization problem (21) can be solved using Algorithm 2.
Let us briefly evaluate the performance of the proposed $\ell$-DER model on several regression datasets from the Penn Machine Learning Benchmarks (PMLB), a significant benchmark suite for machine learning evaluation and comparison [19]. The chosen datasets have small or medium sizes. For simplicity, we fixed the parameters $r_1 = r_2 = 10$ of the $\ell$-DER models for all datasets.

We compared the performance of the $\ell$-DER model trained using the three approaches described in the previous section. The $\ell$-DER trained using the SGD method has implemented using the TensorFlow API (TF). Precisely, because $\ell$-DER is equivalent to a maxout network [10], we used tensorflow-addons, which provides extra functionalities and include the maxout layers. For the other two approaches, we used the CVXPY package [7] with the MOSEK solver [2]. For the DCCP problem, in particular, we used the DCCP extension for the CVXPY available at https://github.com/cvxgrp/dccp.

We would like to point out that we handled missing data using sklearn’s SimpleImputer() command. Furthermore, we partitioned the data set into training and test sets using the sklearn’s StratifiedKFold() command with $k = 5$. Finally, we used the Mean Absolute Percentage Error (MAPE) to measure the performance of the regressor $\ell$-DER for each of the training approaches.

### Table 1. Average and standard deviation of the MAPE Score.

| Datasets (instances, features) | $\ell$-DER | $\ell$-DER | $\ell$-DER |
|-------------------------------|------------|------------|------------|
| Analcatdata_vehicle (48,4)    | 0.2750 ± 0.0495 | 0.2695 ± 0.0358 | 0.1570 ± 0.0529 |
| Bodyfat (252,14)               | 1.2e+14 ± 1.6e+14 | 5.5e+13 ± 3.2e+13 | 2.6e+13 ± 2.8e+13 |
| Cloud (108,5)                 | 2.2e+12 ± 1.8e+12 | 1.2e+12 ± 1.0e+12 | 3.4e+12 ± 2.8e+12 |
| Elusage (55,2)                | 0.1875 ± 0.0361 | 0.2016 ± 0.0152 | 0.1452 ± 0.0145 |
| Pm10 (500,7)                 | 2.2e+12 ± 1.8e+12 | 0.2444 ± 0.0156 | 0.1829 ± 0.0137 |
| Pollen (3848,4)              | 2.1041 ± 0.1791 | 2.0416 ± 0.1623 | 2.0117 ± 0.1660 |
| PwLinear (266)              | 1.0758 ± 0.2917 | 2.1620 ± 0.4312 | 0.3689 ± 0.0611 |
| Rabe_266 (120,2)           | 0.0938 ± 0.0347 | 0.2630 ± 0.0587 | 0.0410 ± 0.0132 |
| Rmftsaldadata (508,10)      | 0.1427 ± 0.0097 | 0.1531 ± 0.0076 | 0.1099 ± 0.0025 |
| Sleuth_cex1605 (62,5)        | 0.0482 ± 0.0077 | 0.0456 ± 0.0014 | 0.0179 ± 0.0076 |
| Vineyard (522)               | 0.0937 ± 0.0019 | 0.0949 ± 0.0035 | 0.0756 ± 0.0059 |
| Visualizing_environmental (111,3) | 0.2278 ± 0.0127 | 0.2301 ± 0.0203 | 0.1856 ± 0.0209 |
| Visualizing_galaxy (323,4)   | 0.0163 ± 0.0010 | 0.6926 ± 1.3183 | 0.0095 ± 0.0052 |

As to the training execution time, the SGD is the fastest procedure. Despite its longer training time, the DCCP is not quite different from the SGD approach. The DCA proved to be slower than the other two approaches. Figure [1] also provides a visual interpretation of the outcome of our computational experiment.
Precisely, the boxplot on Figure 1 has been obtained by normalizing the average MAPE scores produced by the three approaches for each dataset. From the boxplot depicted in Figure 1, the ℓ-DER trained with the DCCP procedure achieved the best performance. The other two approaches proved to be competitive with each other. The non-parametric Wilcoxon hypothesis testing confirmed that the DCCP approach outperformed the other two approaches for training the ℓ-DER, method with a confidence level of 95%.

6 Concluding Remarks

This paper introduced the linear dilation-erosion regressor (ℓ-DER), which is given by a convex combination of the composition of linear transformations and elementary morphological operators. Precisely, an ℓ-DER is defined by the DC function \( \tau^\ell \) given by (13). Because \( \tau^\ell \) is a continuous piecewise linear function, an ℓ-DER is a universal approximator.

In this paper, we trained an ℓ-DER model by minimizing the mean square error given by (17) using training set \( \mathcal{T} = \{ (x_i, y_i) : i = 1 : m \} \subset \mathbb{R}^n \times \mathbb{R} \). Furthermore, we proposed to determine the parameters of the ℓ-DER regressor \( \tau^\ell \) using three different approaches. The first uses the SGD method, while the second approach is based on the DCA subgradient method. The third approach solves a constrained, disciplined convex-concave programming problem for training the ℓ-DER. Both the methods using DCA and DCCP are DC optimization problems.

Finally, we compared the ℓ-DER trained using three approaches using 13 regression tasks. According to the preliminary computational experiments, the DCCP-based approach yielded the best ℓ-DER model in terms of the MAPE score. As to the training time, the SDG-based method outperformed the other two. In the future, we intend to investigate further the performance of the ℓ-DER model. In particular, we plan to compare them with other machine learning models.
References

1. de A. Araújo, R.: A class of hybrid morphological perceptrons with application in time series forecasting. Knowledge-Based Systems 24(4), 513–529 (2011). https://doi.org/10.1016/j.knosys.2011.01.001

2. ApS, M.: MOSEK Optimizer API for Python. - (2020), https://docs.mosek.com/9.2/pythonapi.pdf

3. Araújo, R.d.A., Oliveira, A.L., Soares, S., Meira, S.: An evolutionary morphological approach for software development cost estimation. Neural Networks 32, 285–291 (8 2012). https://doi.org/10.1016/j.neunet.2012.02.040

4. Birkhoff, G.: Lattice Theory. American Mathematical Society, Providence, 3 edn. (1993)

5. Bottou, L.: Large-Scale Machine Learning with Stochastic Gradient Descent. In: Lechevallier, Y., Saporta, G. (eds.) Proceedings of COMPSTAT’2010. pp. 177–186. Physica-Verlag HD, Heidelberg (2010). https://doi.org/10.1007/978-3-7908-2604-3

6. Charisopoulos, V., Maragos, P.: Morphological Perceptrons: Geometry and Training Algorithms. In: Math. Morph. and Its Appli. to Signal and Image Proc. pp. 3–15. Springer (2017). https://doi.org/0.1007/978-3-319-57240-6_1

7. Diamond, S., Boyd, S.: CVXPY: A Python-Embedded Modeling Language for Convex Optimization. - (Jun 2016), https://dl.acm.org/doi/10.5555/2946645.3007036

8. Dinh, T.P., Thi, H.A.L.: Convex analysis approach to d.c. programming: Theory, Algorithm and Applications. Acta Mathematica Vietnamica 22(1), 289 (1997)

9. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)

10. Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., Bengio, Y.: Maxout Networks. In: International Conference on Machine Learning. pp. 1319–1327 (2013)

11. Grant, M., Boyd, S., Ye, Y.: Disciplined Convex Programming. In: Global Optimization, pp. 155–210. Kluwer Academic Publishers (6 2006). https://doi.org/10.1007/0-387-30528-9_7

12. Hartman, P.: On functions representable as a difference of convex functions. Pacific Journal of Mathematics 9(3), 707–713 (Sep 1959), publisher: Mathematical Sciences Publishers

13. Heijmans, H.J.A.M.: Mathematical Morphology: A Modern Approach in Image Processing Based on Algebra and Geometry. SIAM Review 37(1), 1–36 (1995)

14. Hernández, G., Zamora, E., Sossa, H., Téllez, G., Furlán, F.: Hybrid neural networks for big data classification. Neurocomputing 390, 327–340 (5 2020). https://doi.org/10.1016/j.neucom.2019.08.095

15. Ho, V.T., Le Thi, H.A., Pham Dinh, T.: DCA with Successive DC Decomposition for Convex Piecewise-Linear Fitting. In: Le Thi, H.A., Le, H.M., Pham Dinh, T., Nguyen, N.T. (eds.) Advanced Computational Methods for Knowledge Engineering, pp. 39–51. Advances in Intelligent Systems and Computing, Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-38364-0_4

16. Ho, V.T., Le Thi, H.A., Pham Dinh, T.: DCA-based algorithms for DC fitting. Journal of Computational and Applied Mathematics 389, 113353 (Jun 2021). https://doi.org/10.1016/j.cam.2020.113353

17. Lipp, T., Boyd, S.: Variations and extension of the convex-concave procedure. Optimization and Engineering 17(2), 263–287 (Jun 2016). https://doi.org/10.1007/s11081-015-9294-x
Least-Squares Training Linear Dilation-Erosion Regression

18. Oliveira, A.L., Valle, M.E.: Linear Dilation-Erosion Perceptron Trained Using a Convex-Concave Procedure. In: Abraham, A., Ohsawa, Y., Gandhi, N., Jabbar, M., Haqiq, A., McLoone, S., Issac, B. (eds.) Proceedings of SoCPaR 2020. pp. 245–255. Advances in Intelligent Systems and Computing. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-73689-7_24

19. Olson, R.S., La Cava, W., Orzechowski, P., Urbanowicz, R.J., Moore, J.H.: PMLB: a large benchmark suite for machine learning evaluation and comparison. BioData Mining 10(1), 36 (Dec 2017). https://doi.org/10.1186/s13040-017-0154-4

20. Pessoa, L.F.C., Maragos, P.: Neural networks with hybrid morphological/rank/linear nodes: a unifying framework with applications to handwritten character recognition. Pattern Recognition 33, 945–960 (2000)

21. Ritter, G.X., Sussner, P.: An Introduction to Morphological Neural Networks. In: Proceedings of the 13th International Conference on Pattern Recognition. pp. 709–717. Vienna, Austria (1996)

22. Ritter, G.X., Sussner, P., Diaz-De-Leon, J.L.: Morphological associative memories. IEEE Transactions on Neural Networks 9(2), 281–293 (1998). https://doi.org/10.1109/72.661123

23. Ritter, G.X., Urcid, G.: Lattice Algebra Approach to Single-Neuron Computation. IEEE Transactions on Neural Networks 14(2), 282–295 (2003)

24. Shen, X., Diamond, S., Gu, Y., Boyd, S.: Disciplined convex-concave programming. In: 2016 IEEE 55th Conference on Decision and Control (CDC). pp. 1009–1014 (2016). https://doi.org/10.1109/CDC.2016.7798400

25. Soille, P.: Morphological Image Analysis. Springer Verlag, Berlin (1999)

26. Stone, M.H.: The Generalized Weierstrass Approximation Theorem. Mathematics Magazine 21(4), 167–184 (1948). https://doi.org/10.2307/3029750, publisher: Mathematical Association of America

27. Sussner, P., Esmi, E.L.: Morphological Perceptrons with Competitive Learning: Lattice-Theoretical Framework and Constructive Learning Algorithm. Information Sciences 181(10), 1929–1950 (2011). https://doi.org/10.1016/j.ins.2010.03.016

28. Tao, P.D., Souad, E.B.: Duality in D.C. (Difference of Convex functions) Optimization. Subgradient Methods. In: Hoffmann, K.H., Zowe, J., Hiriart-Urruty, J.B., Lemarechal, C. (eds.) Trends in Mathematical Optimization: 4th French-German Conference on Optimization, pp. 277–293. International Series of Numerical Mathematics, Birkhäuser, Basel (1988). https://doi.org/10.1007/978-3-0348-9297-1_18

29. Tuy, H.: DC Functions and DC Sets. In: Tuy, H. (ed.) Convex Analysis and Global Optimization, pp. 103–123. Springer Optimization and Its Applications, Springer International Publishing, Cham (2016). 10.1007/978-3-319-31484-6_4

30. Valle, M.E.: Reduced Dilation-Erosion Perceptron for Binary Classification. Mathematics 8(4), 512 (2020). https://doi.org/10.3390/math8040512

31. Wang, S.: General constructive representations for continuous piecewise-linear functions. IEEE Transactions on Circuits and Systems I: Regular Papers 51(9), 1889–1896 (2004). https://doi.org/10.1109/TCSI.2004.834521

32. Yuille, A.L., Rangarajan, A.: The concave-convex procedure. Neural Computation 15(4), 915–936 (2003). https://doi.org/10.1162/08997660360581958