DC Neural Networks avoid overfitting in one-dimensional nonlinear regression

Cesar Beltran-Royo[a,*], Laura Llopis-Ibor[a], Juan J. Pantrigo[a], Iván Ramírez[a]

[a]*Computer Science and Statistics Department, Universidad Rey Juan Carlos, C. Tulipán, s/n, 28933 Móstoles, Madrid, Spain*

Abstract

In this paper, we analyze Difference of Convex Neural Networks in the context of one-dimensional nonlinear regression. Specifically, we show the surprising ability of the Difference of Convex Multilayer Perceptron (DC-MLP) to avoid overfitting in nonlinear regression. Otherwise said, DC-MLPs self-regularize (do not require additional regularization techniques). Thus, DC-MLPs could result very useful for practical purposes based on one-dimensional nonlinear regression. It turns out that shallow MLPs with a convex activation (ReLU, softplus, etc.) fall in the class of DC-MLPs. On the other hand, we call SQ-MLP the shallow MLP with a Squashing activation (logistic, hyperbolic tangent, etc.). In the numerical experiments, we show that DC-MLPs used for nonlinear regression avoid overfitting, in contrast with SQ-MLPs. We also compare DC-MLPs and SQ-MLPs from a theoretical point of view.

Keywords: DC neural network, multilayer perceptron, nonlinear regression, overfitting.

1. Introduction

Neural Networks (NNs) have proven great success in the AI field, ranging from Natural Language Processing tasks, as speech recognition [1, 2] and language representation and understanding [3, 4], to Computer Vision problems as

*Corresponding author

Email addresses: cesar.beltran@urjc.es (Cesar Beltran-Royo), laura.llopis@urjc.es (Laura Llopis-Ibor), juanjose.pantrigo@urjc.es (Juan J. Pantrigo), ivan.rammirez@urjc.es (Iván Ramírez)

image classification and recognition [5, 6], image generation [7] and other tasks such as forecasting regression problems [8, 9].

As universal approximators [10, 11], NNs have shown particularly good generalization properties in high dimensional data, tackling and overcoming -to some extent-, the well-known *curse of dimensionality* problem coined by Richard Bellman in 1966 [12]. One major drawback, in practice, is that highly parameterized neural networks (neural networks with high capacity) are prone to overfit the training data, performing poorly in test examples. Recently and contrary to previous assumptions, new findings have revealed that increasing the number of parameters in neural networks can actually improve their generalization performance. This phenomenon, known as double-descent [13], suggests that the increase in parameter count induces a form of regularization in the solution. This explanation partially accounts for the improved performance observed in the presence of a larger number of parameters. These results encourage further studies on how neural networks actually behave and how to induce more regularizing properties. Training such networks is usually performed through gradient-based optimization methods, leading to suboptimal results (local minima) given that, in this context, loss functions are nonconvex [14, 15].

Convexity in neural networks has been a topic of interest since the beginning of the deep learning era [16]. In this context, convexity typically refers to the convexity analysis of the loss function (w.r.t. the training parameters). Recently, the convexity of NNs w.r.t. the input variables have also been studied under the name Input Convex Neural Networks (ICNN) [17]. Although ICNNs have interesting properties, they can only be used to learn convex functions. That is, the convexity assumption greatly limits the model's capacity. Thus, in order to learn nonconvex functions, ICNNs have to be combined with other tools. For example, in [18] the CIFAR-100 classification problem, a nonconvex problem, is tackled by an ensemble of ICNNs combined with an auxiliary shallow MLP (a nonconvex function w.r.t. the input variables).

Also in the context of convexity, [19] takes a step forward by proposing the Convex Difference Neural Networks (CDiNN). Specifically, CDiNN are neural networks which can be decomposed as the Difference of two Convex functions (DC). It is well known that DC functions can approximate, to a given numerical tolerance, virtually any function (convex or nonconvex) [20]. Therefore, CDiNNs, on the one hand, can learn virtually any function, in contrast to ICNNs, and on the other hand, have a convexity-based structure as a DC function. CDiNNs were introduced in [19] as a new NN architecture mainly for efficient decision-making without significant loss of representational capability. More specifically, these

authors applied DC optimization algorithms [21] to decision-making involving NNs.

In contrast to [19], we apply CDiNNs for nonlinear regression. Nonlinear regression problems can be solved by different machine learning approaches such as neural networks, support vector regression (SVR), decision trees and random forest. Usually, one uses the corresponding state of the art software, as for example, the popular XGBoost [22], an open-source library which implements a decision-tree based ensemble algorithm that uses a gradient boosting framework. Such approaches are used in a wide range of applications, as for example, fault diagnosis [23], time series forecasting [24] or biomedical signal estimation [25].

In this paper we focus on nonlinear regression based on NNs [26]. An overview of regression algorithms, among them the NN approach, can be found in [27]. In order to improve the effectiveness of nonlinear regression based on NNs, different aspects have been studied. For example: a) The estimation of the regularization term used to avoid overfitting in the training process can be found in [28]. b) The robustness of NNs in regression tasks is improved in [29], where a prediction interval and a point prediction are simultaneously generated by means of a deep NN. c) Another example corresponds to building an explicit expression for the coefficients of a polynomial regression from the weights of a given NN, using a Taylor expansion approach [30], etc.

The gap between theory and practice in deep learning is narrowing in recent years. Thus, for example, [31] studies deep learning through the prism of interpolation in order to get closer to a general theory of deep learning. In this context, we aim at shedding light upon the surprising ability of CDiNNs to avoid overfitting in nonlinear regression, despite the fact that they remain universal approximators [19]. To this end, we restrict ourselves to DC shallow Multilayer Perceptron (DC-MLP) architectures as one-dimensional nonlinear regression functions. We analyze their properties, with a special focus on the derivative w.r.t. the input variable. Roughly speaking, DC-MLPs show low derivative values w.r.t. the input variable which is typical, as is shown in this work, in functions without overfitting (see Fig. 2). This will play a similar role to common regularization techniques such as weight-decay (L_2-regularization) or dropout [32], among others.

However, some fields, as in Physics or Control Theory, require more than a straightforward implementation of a NN to get some of their advantages. In particular, the derivative of the NN w.r.t. the input is crucial to characterize the system. For instance, in Physics Informed Neural Networks (PINNs) [33], neural networks are adapted to provide control over the first order derivatives in order to match the physical constrains. In [34], the authors exploit sector-bounded and

finite slope activation functions to provide stability guarantees for NNs approximated controllers. Recently, residual skip connections [6] have been cast as a discretized version of Neural ODEs [35], where the residual matches the derivative of the NN (layerwise).

Furthermore, the derivative –w.r.t. the input– is directly related to a whole new subject of study, so-called adversarial examples [36, 37]. They exploit the derivative of a NN to create examples that drastically flip the final prediction of the network. As a consequence, NNs are called into doubt in terms of explainability-interpretability and generalization. Regardless their motivation, the analysis of the derivative of NNs is gaining momentum; reason why, in this paper, we opt to focus on the nonlinear regression problem to analyze and compare DC-MLPs and SQ-MLPs (Squashing Multilayer Perceptrons) through their derivatives.

The contribution of this paper is twofold:

a) From an empirical point of view we show that DC-MLPs, used for one-dimensional nonlinear regression, avoid overfitting in contrast with SQ-MLPs (see Fig. 1). Furthermore, the overfitting level of SQ-MLPs results directly proportional to: the number of neurons, the number of training iterations and the magnitude of the data noise. However, it decreases as the number of data points increases.

b) From a theoretical point of view we prove: i) That the derivative of the rectified MLP, the typical DC-MLP, has a moderate variation. ii) That the derivative of the logistic MLP, the typical SQ-MLP, is potentially unbounded. iii) That overfitting regression functions have a potentially unbounded derivative. These theoretical results would explain, at least partially, the empirical results in a).

This paper is organized as follows. Section 2 is devoted to analyze the derivative of one-dimensional nonlinear regression functions: The general case is studied in Section 2.1, the logistic MLP case in Section 2.2 and the rectified MLP case in Section 2.4. In Section 3 we develop an exhaustive experimental study, focusing on the surprising ability of DC-MLPs to avoid overfitting in nonlinear regression: Section 3.1 presents a set of experiments, based on synthetic data, to compare the effect of parameter setting and data characteristics on the performance of DC-MLPs versus SQ-MLPs. In Section 3.2 we carry out an additional set of experiments based on real-world data. Finally, Section 4 summarizes the main conclusions obtained from this research work.

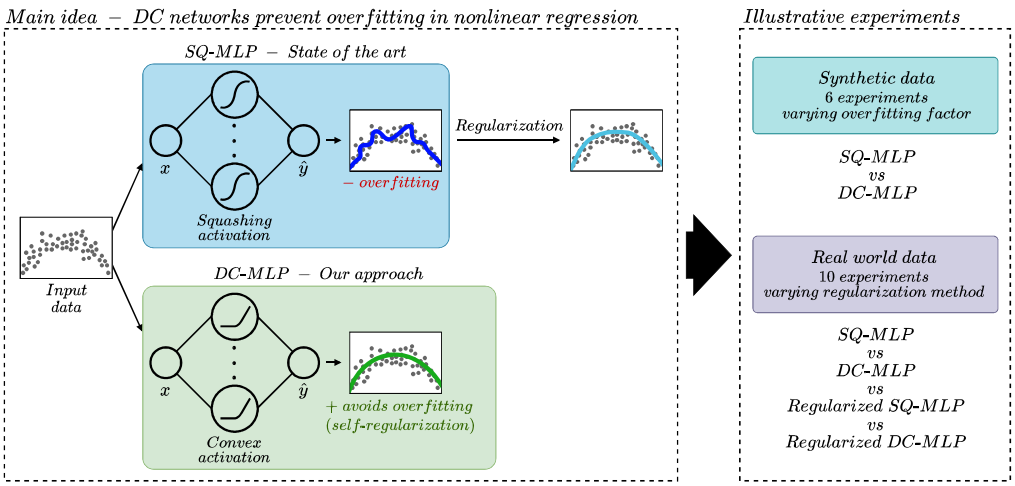

Figure 1: DC-MLPs, used for one-dimensional nonlinear regression, avoid overfitting in contrast with SQ-MLPs.

2. One-dimensional nonlinear regression

Let us consider the following one-dimensional statistical model:

$$Y^{(i)} = g(x^{(i)}) + \varepsilon^{(i)} \qquad \varepsilon^{(i)} \sim N(0, \sigma^2), \text{ for all } i \in \mathcal{I} = \{1, \ldots, I\}, \tag{1}$$

where $g(x) : \mathbb{R} \to \mathbb{R}$ is an unknown nonlinear continuous function, $Y^{(i)}$ is the response or dependent random variable, $x^{(i)}$ is the independent variable and $\varepsilon^{(i)}$ is the error, which is assumed to be a normal random variable for all $i \in \mathcal{I}$. The objective of nonlinear regression is to approximate the unknown function $g(x)$ by means of a regression function, say $f(x; \theta)$.

In order to approximate $g(x)$ by $f(x; \theta)$, one usually collects a sample o data set:

$$\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i \in \mathcal{I}} \subset \mathbb{R}^2,$$

which is used to find the best-fitting parameters θ^* by minimizing a given loss function, say $L(\theta)$, typically the mean squared error (MSE):

$$\min L(\theta) = \frac{1}{I} \sum_{i \in \mathcal{I}} \big(y^{(i)} - f(x^{(i)}; \theta) \big)^2. \tag{2}$$

In nonlinear regression, the solution of the MSE problem (2) is usually performed by numerical optimization algorithms and the whole minimization process is often

termed 'training' the regression function. Thus, the resulting regression function to be used for practical purposes is $f(x;\theta^*)$. Throughout this paper we make the following assumptions:

Assumption 1. *We assume noisy observations:*

$$y^{(i)} = g^{(i)} + e^{(i)} \qquad \text{for all } i \in \mathcal{I},$$

where for a compact notation we have defined $g^{(i)} = g(x^{(i)})$. Furthermore, the errors $\{e^{(i)}\}_{i\in\mathcal{I}}$ are realizations of the independent random variables $\{\varepsilon^{(i)}\}_{i\in\mathcal{I}}$. Consequently, $y^{(i)}$ is a realization of the random variable $Y^{(i)} \sim N(g^{(i)}, \sigma^2)$, for all $i \in \mathcal{I}$.

Assumption 2. *For any data set $\mathcal{D}$ in this paper, we assume that $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i\in\mathcal{I}}$, and that*

$$-\infty \quad < \quad A = x^{(1)} < x^{(2)} < \ldots < x^{(I)} = B \quad < \quad +\infty$$

is a sequence of equidistant points, that divides the regression interval $[A, B]$ into $I-1$ subintervals $[x^{(i)}, x^{(i+1)}]$ of length $L = (B-A)/(I-1)$.

Definition 1 (Shallow MLP)**.** *We define the shallow MLP as the following two-layer MLP with one-dimensional input and output*

$$f(x;\theta) = d + \sum_{j\in\mathcal{J}} c_j \; s(a_j x + b_j), \tag{3}$$

where $\mathcal{J} = \{1, \ldots, J\}$ is the index set, $\theta = (a_j, b_j, c_j, d)_{j\in\mathcal{J}}$ is the vector of training parameters, $s(\cdot)$ is the nonlinear activation function of the hidden layer and the identity is the activation function of the output layer.

If $s(\cdot)$ is the logistic activation, then we will call it a logistic shallow MLP or, for short, a logistic MLP. Analogously, we define softplus MLP, rectified MLP, etc.

Assumption 3. *All the NNs considered in this paper correspond to shallow MLPs (except in Appendix A, where we also consider deep MLPs).*

Assumption 4. *We denote by $f'(x;\theta)$ the derivative of f with respect to the input variable x. Furthermore, all the derivatives considered in this paper are respect to x.*

2.1. Overfitting regression functions have a potentially unbounded derivative

In this section we show that the derivative of overfitting regression functions is potentially unbounded on regression intervals with large data sets as in Fig. 2 (C).

Definition 2 (Overfitting regression function)**.** *Given θ^*, a fixed vector of regression parameters, we say that the regression function $f(x;\theta^*)$ is an overfitting function with tolerance $\epsilon > 0$ for the data set $\mathcal{D}$, if there exist $\mid r^{(i)} \mid < \epsilon$ such that*

$$f(x^{(i)};\theta^*) = y^{(i)} + r^{(i)} \qquad \text{for all } i \in \mathcal{I}.$$

The previous definition corresponds to a 'full' overfitting function, where the regression function 'learns' all the data points (Fig. 2 (A)). Although this concept is useful for the theoretical analysis of the following sections, in applications one usually faces 'partial' overfitting regression functions (Fig. 2 (B) and (C)). Also, in Fig. 2 (C) we observe that the derivative of overfitting regression functions can be very large when trained with large data sets, such that the corresponding graph is zigzag-shaped in some sections.

The following lemma will be used to prove Proposition 1.

Lemma 1. *Given a normal random variable $X \sim N(\mu, \sigma^2)$ then*

$$\mathbb{E}\big[\ \mid X \mid\ \big] \quad \geq \quad 2\,\Phi\big(\,\frac{-1-\mu}{\sigma}\,\big).$$

Proof. If $h(\cdot)$ is the probability density function of X, then

$$\begin{aligned}
\mathbb{E}\big[\ \mid X \mid\ \big] &= \int_{-\infty}^{+\infty} \mid x \mid\ h(x)dx \\
&= \int_{|x|\leq 1} \mid x \mid\ h(x)dx + \int_{|x|>1} \mid x \mid\ h(x)dx \\
&\geq \int_{|x|>1} \mid x \mid\ h(x)dx \quad \geq \quad \int_{|x|>1} 1\, h(x)dx \\
&= P\big(\ \mid X \mid > 1\,\big) \quad = \quad 2P\big(\,X < -1\,\big) \\
&= 2P\big(\,Z < \frac{-1-\mu}{\sigma}\,\big) \quad = \quad 2\Phi\big(\,\frac{-1-\mu}{\sigma}\,\big),
\end{aligned}$$

where $Z \sim N(0,1)$ and $\Phi(\cdot)$ is its probability distribution function. ∎

In the following proposition we see that the expectation of the derivative of an overfitting function $f(x;\theta_I)$ is unbounded along the regression interval $[A, B]$ as I, the number of data points, increases (Fig. 1 (C)). Notice that $f'(x;\theta_I)$ is a random variable since it depends on the realization of the data set $\mathcal{D}_I$.

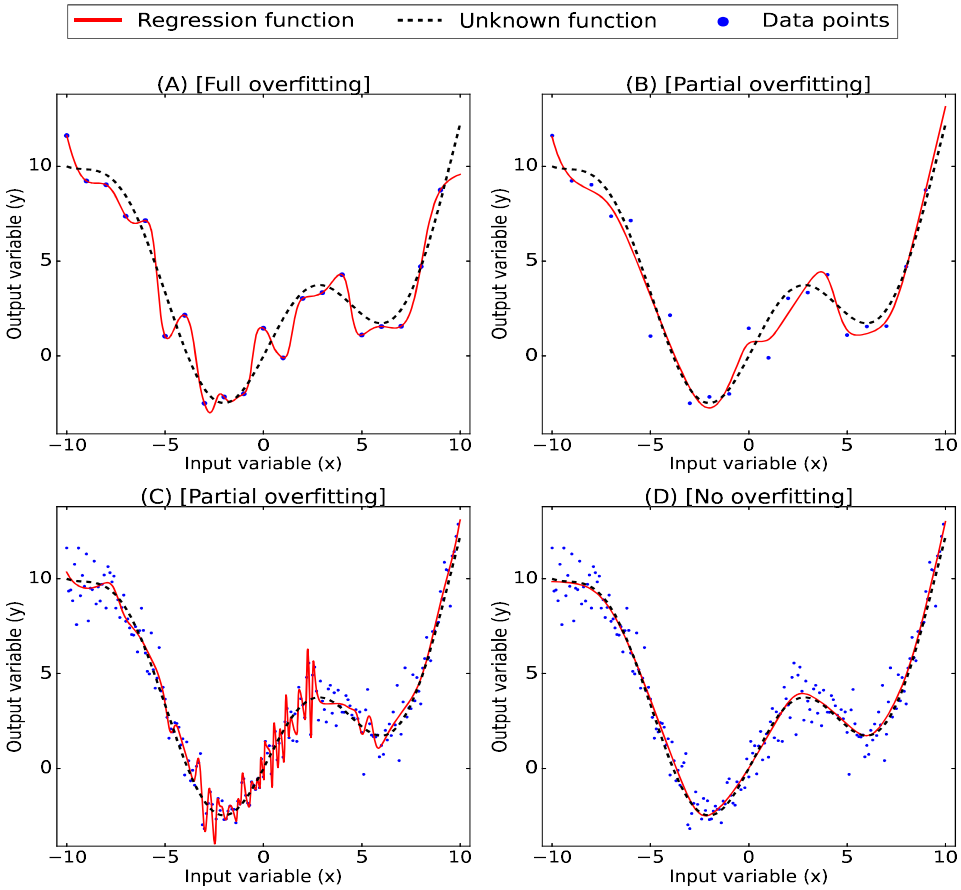

Figure 2: Regression functions can exhibit different levels of overfitting. (A) Full overfitting: The regression function 'learns' all the data points. (B) Partial overfitting (mild). (C) Partial overfitting (strong): The derivative of overfitting regression functions can be very large when trained with large data sets. The corresponding graph is zigzag-shaped in some sections. (D) No overfitting: The ideal regression function replicates the unknown function.

Proposition 1. *If $f(x;\theta_I)$ is an overfitting regression function for $\mathcal{D}_I = \{(x_I^{(i)}, y_I^{(i)})\}_{i\in\mathcal{I}}$, for all $I \geq 2$, then each subinterval $(x_I^{(i)}, x_I^{(i+1)})$ has a point $t_I^{(i)}$ such that*

$$\lim_{I\to+\infty} \mathbb{E}\big[\ |\ f'(t_I^{(i)};\theta_I)\ |\ \big] = +\infty \qquad \textit{for all } i \in \mathcal{I}^{\,-} = \{1,\ldots,I-1\}.$$

Proof. This proof has four steps:
Step 1) Given any fixed $I \geq 2$, then by the mean-value Theorem 5.11 [38], there exists $t_I^{(i)} \in (x_I^{(i)}, x_I^{(i+1)})$ such that

$$f'(t_I^{(i)};\theta_I) = \frac{y_I^{(i+1)} - y_I^{(i)}}{x_I^{(i+1)} - x_I^{(i)}} \qquad \text{for all } i \in \mathcal{I}^{\,-}.$$

Prior observing $y_I^{(i)}$ and $y_I^{(i+1)}$, all we know about the data is summarized by the random variables $Y_I^{(i)}$ and $Y_I^{(i+1)}$. Thus prior observing $y_I^{(i)}$ and $y_I^{(i+1)}$, one can write:

$$f'(t_I^{(i)};\theta_I) = \frac{Y_I^{(i+1)} - Y_I^{(i)}}{x_I^{(i+1)} - x_I^{(i)}} \qquad \text{for all } i \in \mathcal{I}^{\,-}.$$

Notice that $f'(t_I^{(i)};\theta_I)$, being a function of the random variables $Y_I^{(i+1)}$ and $Y_I^{(i)}$, is also a random variable. This implies that

$$|\ f'(t_I^{(i)};\theta_I)\ | = \frac{|\ Y_I^{(i+1)} - Y_I^{(i)}\ |}{|\ x_I^{(i+1)} - x_I^{(i)}\ |}.$$

Thus

$$\mathbb{E}\big[\ |\ f'(t_I^{(i)};\theta_I)\ |\ \big] = \frac{\mathbb{E}\big[\ |\ Y_I^{(i+1)} - Y_I^{(i)}\ |\ \big]}{L_I} \qquad \text{for all } i \in \mathcal{I}^{\,-},$$

where $L_I = (b-a)/(I-1)$.

Step 2) Let us see the distribution of $Y_I^{(i+1)} - Y_I^{(i)}$. By the hypothesis, we have that

$$Y_I^{(i)} = g(x_I^{(i)}) + \epsilon^{(i)} \sim N(g(x_I^{(i)}), \sigma^2) \qquad \text{for all } i \in \mathcal{I}.$$

Given that $Y_I^{(i)}$ and $Y_I^{(i+1)}$ are independent normal random variables:

$$Y_I^{(i+1)} - Y_I^{(i)} \sim N(\mu_I^{(i)}, (\sigma_I^{(i)})^2) \qquad \text{for all } i \in \mathcal{I}^{\,-},$$

with $\mu_I^{(i)} = g(x_I^{(i+1)}) - g(x_I^{(i)})$ and $(\sigma_I^{(i)})^2 = 2\sigma^2$.

Step 3) Let us see that $\lim_{I\to+\infty} \mathbb{E}\big[\mid Y_I^{(i+1)} - Y_I^{(i)} \mid \big]$ is lower bounded. By Lemma 1 we have:

$$\mathbb{E}\big[\mid Y_I^{(i+1)} - Y_I^{(i)} \mid \big] \quad \geq \quad 2\Phi(\frac{-1-\mu_I^{(i)}}{\sigma_I^{(i)}}).$$

Then

$$\lim_{I\to+\infty} \mathbb{E}\big[\mid Y_I^{(i+1)} - Y_I^{(i)} \mid \big] \quad \geq \quad \lim_{I\to+\infty} 2\Phi(\frac{-1-\mu_I^{(i)}}{\sigma_I^{(i)}}) \quad = \quad 2\Phi(\frac{-1}{\sigma\sqrt{2}}) \quad = \quad K.$$

where we have used that

$$\lim_{I\to+\infty} \mu_I^{(i)} = \lim_{I\to+\infty} \big(g(x_I^{(i+1)}) - g(x_I^{(i)}) \big) = 0,$$

since $\lim_{I\to+\infty} \mid x_I^{(i+1)} - x_I^{(i)} \mid = 0$ and, by hypothesis, g is continuous.

Step 4) Let us compute the limit of the expected derivative

$$\begin{aligned} \lim_{I\to+\infty} \mathbb{E}\big[\mid f'(t_I^{(i)};\theta_I) \mid \big] \quad &= \quad \lim_{I\to+\infty} \frac{\mathbb{E}\big[\mid Y_I^{(i+1)} - Y_I^{(i)} \mid \big]}{L_I} \\ &= \quad +\infty \qquad\qquad \text{for all } i \in \mathcal{I}^-, \end{aligned}$$

since the limit of the numerator is lower bounded by $K > 0$ (Step 3) and the limit of the denominator is 0. ■

2.2. Logistic MLPs have a potentially unbounded derivative

The objective of this paper is to compare SQ-MLPs versus DC-MLPs, as one-dimensional regression functions to asses their capacity to avoid overfitting. Squashing MLPs correspond to shallow MLPs defined by equation (3) with an activation function $s(\cdot)$ of type squashing (logistic, hyperbolic tangent, softsign, etc.). We denote them SQ-MLPs. On the other hand, DC-MLPs will be introduced in Section 2.3. One of the prototypical SQ-MLPs is the logistic MLP, which corresponds to the shallow MLP with a logistic activation, i.e.,

$$s(t) = \sigma(t) = 1/(1+\exp(-x)).$$

In this section we analyze the logistic MLP as a nonlinear regression function. Notice that for simplicity this section focus on logistic MLPs, but in Section 3 we will see that the performance obtained by other SQ-MLPs is very similar.

The following lemma will be used to prove Proposition 2.

Lemma 2. *Given any point $\bar{x} \in \mathbb{R}$ then:*

$$\lim_{\alpha \to 0^+} \sigma\big(\frac{x-\bar{x}}{\alpha}\big) = \begin{cases} 0 & \textit{if} \quad x < \bar{x} \\ 0.5 & \textit{if} \quad x = \bar{x} \\ 1 & \textit{if} \quad x > \bar{x}, \end{cases}$$

where $\lim_{\alpha \to 0^+}$ denotes the right one-sided limit.

Proof. The proof is direct taking into account the definition of $\sigma(t)$. ∎

In the context of NNs with one-dimensional input, [39] shows that the shallow MLP with logistic activation can potentially learn any continuous function on a compact interval. In order to complement the previous result in the context of nonlinear regression, next proposition shows that: a) The shallow MLP with logistic activation can potentially suffer overfitting for any regression data set. b) Furthermore, its derivative w.r.t the input variable is potentially unbounded on the regression interval, as is the case of overfitting regression functions trained with large data sets (see Proposition 1). In the experiments of Section 3 we will see that logistic MLPs tend to produce overfitting.

Proposition 2. *Let us consider the data set $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i \in \mathcal{I}}$ and the logistic MLP $f(x;\theta)$ with J neurons $(J = I)$. Then, for any $\epsilon > 0$ and any $M > 0$ there exist $\theta^* = \big(a_j^*, b_j^*, c_j^*, d^*\big)$ and $\mid r^{(i)} \mid < \epsilon$ such that:*

$$a) \quad f(x^{(i)};\theta^*) = y^{(i)} + r^{(i)} \qquad \textit{for all } i \in \mathcal{I}. \tag{4}$$

$$b) \quad \mid f'(x^{(i)};\theta^*) \mid > M \qquad \textit{for all } i \in \widetilde{\mathcal{I}} = \{i \in \mathcal{I} : c_i^* \neq 0\}. \tag{5}$$

Proof. This proof can be structured into three steps:
Step 1) In this step we see that there is a vector of parameters θ_1^* that fulfills statement a) for any given $\epsilon > 0$. Let us consider the following coefficients parameterized by $\alpha > 0$:

$$\begin{aligned} a_j(\alpha) &= \frac{1}{\alpha} && \text{for all } j \in \mathcal{J} \\ b_j(\alpha) &= \frac{-x^{(j)}}{\alpha} \\ c_j(\alpha) &= c_j^* \\ d(\alpha) &= 0, \end{aligned}$$

such that parameters c_j^* fulfill the following system of linear equations (notice that this system can be solved easily):

$$\begin{aligned} y^{(1)} &= \frac{1}{2}c_1^* \\ y^{(2)} &= c_1^* + \frac{1}{2}c_2^* \\ &\dots \\ y^{(I)} &= c_1^* + c_2^* + \ldots + \frac{1}{2}c_I^*, \end{aligned}$$

that is,

$$y^{(i)} = \sum_{j<i} c_j^* + \frac{1}{2}c_i^* \qquad\qquad i \in \mathcal{I}. \tag{6}$$

In this case we can define $\theta(\alpha) = \big(a_j(\alpha), b_j(\alpha), c_j(\alpha), d(\alpha) \big)_{j\in\mathcal{J}}$ such that

$$f(x;\theta(\alpha)) = \sum_{j\in\mathcal{J}} c_j^*\, \sigma\big(\frac{x - x^{(j)}}{\alpha}\big). \tag{7}$$

Notice that $f(x;\theta(\alpha))$, as a function of α, is a continuous function for any $\alpha > 0$ since it is a linear combination of (continuous) logistic functions. Then, given any fixed $x^{(i)}$ from the data set we have

$$\begin{aligned} \lim_{\alpha\to 0^+} f(x^{(i)};\theta(\alpha)) &= \lim_{\alpha\to 0^+} \sum_{j<i} c_j^*\, \sigma\big(\frac{x^{(i)} - x^{(j)}}{\alpha}\big) \\ &\quad + \lim_{\alpha\to 0^+} \sum_{j=i} c_j^*\, \sigma\big(\frac{x^{(i)} - x^{(j)}}{\alpha}\big) \\ &\quad + \lim_{\alpha\to 0^+} \sum_{j>i} c_j^*\, \sigma\big(\frac{x^{(i)} - x^{(j)}}{\alpha}\big) \\ &= \sum_{j<i} c_j^* + \frac{1}{2}c_i^* + 0 \qquad\qquad \text{(by Lemma 2)} \\ &= y^{(i)}. \qquad\qquad \text{(by equation (6))} \end{aligned}$$

Therefore, given that $f(x;\theta(\alpha))$ as a function of α is continuous, then for any $\epsilon > 0$ there exists $\delta_1 > 0$ such that for all $\alpha_1^* \in (0, \delta_1)$ we have that

$$|\, y^{(i)} - f(x^{(i)};\theta_1^*)\, | < \epsilon \qquad\qquad \text{for all } i \in \mathcal{I},$$

where $\theta_1^* = \theta(\alpha_1^*)$.

Step 2) In this step we see that there is a vector of parameters θ_2^* that fulfills statement b) for any given $M > 0$. First, let us compute the derivative of $f(x;\theta)$ defined by equation (7). Given that $\sigma'(\cdot) = \sigma(\cdot) - \sigma(\cdot)^2$ we have:

$$f'(x;\theta(\alpha)) = \frac{1}{\alpha}\sum_{j\in\mathcal{J}} c_j^* \,\big(\,\sigma\big(\,\frac{x - x^{(j)}}{\alpha}\,\big) \;- \sigma\big(\,\frac{x - x^{(j)}}{\alpha}\,\big)^2\,\big).$$

Second, let us compute the following limit for any fixed $x^{(i)}$:

$$\lim_{\alpha\to 0^+} f'(x^{(i)};\theta(\alpha)) = \lim_{\alpha\to 0^+} \frac{1}{\alpha}\sum_{j\in\mathcal{J}} c_j^* \,(\sigma_{ij} \;- \sigma_{ij}^2),$$

where

$$\sigma_{ij} = \sigma\big(\,\frac{x^{(i)} - x^{(j)}}{\alpha}\,\big).$$

That limit can be decomposed as follows

$$\lim_{\alpha\to 0^+} \frac{1}{\alpha}\Big(\sum_{j<i} c_j^* \,(\sigma_{ij} \;- \sigma_{ij}^2) + \sum_{j=i} c_j^* \,(\sigma_{ij} \;- \sigma_{ij}^2) + \sum_{j>i} c_j^* \,(\sigma_{ij} i \;- \sigma_{ij}^2)\Big).$$

The limit of the numerator can be computed as follows:

$$\begin{aligned}
&\lim_{\alpha\to 0^+} \Big(\sum_{j<i} c_j^* \,(\sigma_{ij} \;- \sigma_{ij}^2) + \sum_{j=i} c_j^* \,(\sigma_{ij} \;- \sigma_{ij}^2) + \sum_{j>i} c_j^* \,(\sigma_{ij} \;- \sigma_{ij}^2)\Big) \\
&= 0 + c_i^*\big(\,\frac{1}{2} - \frac{1}{4}\,\big) + 0 \qquad \text{(by Lemma 2)} \\
&= \frac{c_i^*}{4}.
\end{aligned}$$

Thus

$$\lim_{\alpha\to 0^+} f'(x^{(i)};\theta(\alpha)) = \lim_{\alpha\to 0^+} \frac{c_i^*}{4\,\alpha} = \begin{cases} -\infty & \text{if} \quad c_i^* < 0 \\ 0 & \text{if} \quad c_i^* = 0 \\ +\infty & \text{if} \quad c_i^* > 0. \end{cases}$$

Therefore, for any $M > 0$ there exists $\delta_2 > 0$ such that for all $\alpha_2^* \in (0, \delta_2)$ we have

$$\mid f'(x^{(i)};\theta_2^*) \mid > M \qquad\qquad \text{for all } i \in \widetilde{\mathcal{I}},$$

where $\theta_2^* = \theta(\alpha_2^*)$.

Step 3) To finish the prove, let us see that we can take a common θ^* that fulfills statements a) and b) simultaneously, for any given $\epsilon > 0$ and $M > 0$. It is enough to take $\delta = \min\{\delta_1, \delta_2\}$, where δ_1 and δ_2 have been defined in steps 1) and 2), respectively. In this case, equations (4) and (5) are fulfilled simultaneously for any $\alpha^* \in (0, \delta)$. Therefore one can take $\theta^* = \theta(\alpha^*)$. ■

2.3. Difference of convex shallow MLPs

Difference of Convex (DC) neural networks were introduced in [19] as a new NN architecture mainly for efficient decision making without significant loss of representational capability. More specifically these authors applied DC optimization techniques to decision making involving NNs. In contrast to [19], we use DC neural networks to avoid overfitting in nonlinear regression. We restrict ourselves to the case of DC shallow MLPs which we call DC-MLPs and define as follows.

Definition 3 (DC-MLP)**.** *Given the shallow MLP*

$$f(x;\theta) = d + \sum_{j\in\mathcal{J}} c_j \; s(a_j x + b_j),$$

we say that it is a DC-MLP if it can be decomposed as follows:

$$\begin{aligned} f(x;\theta) &= f_1(x;\theta_1) + f_2(x;\theta_2) \\ f_1(x;\theta_1) &= \frac{d}{2} + \sum_{j\in\mathcal{J}_1} c_j \; s(a_j x + b_j) \\ f_2(x;\theta_2) &= \frac{d}{2} + \sum_{j\in\mathcal{J}_2} c_j \; s(a_j x + b_j), \end{aligned}$$

where $\mathcal{J} = \mathcal{J}_1 \cup \mathcal{J}_2$ is a partition of the index set, $f_1(x;\theta_1)$ and $f_2(x;\theta_2)$ are convex and concave functions of x, respectively, and $\theta = (\theta_1, \theta_2)$ with $\theta_1 = (a_j, b_j, c_j, d)_{j\in\mathcal{J}_1}$ and $\theta_2 = (a_j, b_j, c_j, d)_{j\in\mathcal{J}_2}$.

Notice that the DC-MLP is the simplest version of the CDiNN (Convex Difference Neural Network) defined in [19], since CDiNNs can be arbitrarily deep with multidimensional input and output.

In Proposition 3 we show that DC-MLPs are DC functions [20]. In Proposition 4 we prove that any shallow MLP with convex activation is a DC-MLP. Finally, in Section 2.4 we analyze the derivative of the rectified MLP, one of the prototypical DC-MLPs.

Proposition 3. *If $f(x;\theta)$ is a DC-MLP, then it is a DC function on $\mathbb{R}$.*

Proof. It is enough to write $f(x;\theta) = f_1(x;\theta_1) - \big(- f_2(x;\theta_2) \big)$ to show that $f(x;\theta)$ can be expressed as the difference of two convex functions on x, for all $x \in \mathbb{R}$. ∎

Proposition 4. *If $f(x;\theta)$ is a shallow MLP with a nonlinear activation function which is convex (softplus, ReLU, leaky ReLU, ELU, etc.), then it is a DC-MLP.*

Proof. It is enough to take $\mathcal{J}_1 = \{j \in \mathcal{J} \mid c_j > 0\}$ and $\mathcal{J}_2 = \{j \in \mathcal{J} \mid c_j < 0\}$ to fulfill definition 3. In this case

$$f_1(x;\theta_1) = \frac{d}{2} + \sum_{j \in \mathcal{J}_1} c_j \; s(a_j x + b_j)$$

is a convex function of x since it is an affine combination of convex functions with positive weights c_j (notice that $s(a_j x + b_j)$ is convex for all $j \in \mathcal{J}_1$ [40]). Furthermore, it can be seen in the same way that $f_2(x;\theta_2)$, defined analogously, is a concave function of x. ∎

From the previous proposition, there exist different ways to implement DC-MLPs. As an example, we provide a minimal pseudo-code implementing a DC-MLP using PyTorch [41] in Appendix A.

2.4. Rectified MLPs have a derivative with moderate variation

As already pointed out, we aim to asses the capacity of SQ-MLPs and DC-MLPs to avoid overfitting in one-dimensional nonlinear regression. One of the prototypical DC-MLPs is the rectified MLP, which corresponds to the shallow MLP defined by equation (3) with a rectified activation, i.e.,

$$s(t) = ReLU(t) = \max\{0, t\}.$$

In this section we analyze the derivative of the rectified MLP. Notice that for simplicity this section focus on rectified MLPs, but in Section 3 we will see that the performance obtained by other DC-MLPs (ELU, softplus,) is very similar.

The following lemma will be used to prove Proposition 5.

Lemma 3. *Consider the regression interval $[A, B]$, and the functions:*

$$\varphi_1(x) = ReLU(ax + b) \qquad \textit{with } a < 0$$
$$\varphi_2(x) = ReLU(-ax - b) - ReLU(-ax + aA) + \varphi_1(A),$$

where we assume that $\varphi_1(x)$ is not constantly zero in $[A, B]$. Then $\varphi_1(x) = \varphi_2(x)$ for all $x \in [A, B]$.

Proof. Given that $a < 0$, then

$$\varphi_1(x) = \begin{cases} ax + b & \text{if} \quad x \leq -b/a \\ 0 & \text{if} \quad x > -b/a. \end{cases}$$

Therefore, it is clear that if $\varphi_1(x)$ is not constantly zero in $[A, B]$, then $-b/a > A$. To proof this lemma, we distinguish two cases.

Case 1: $A < -b/a < B$. In this case is is enough to see that $\varphi_1(x) = \varphi_2(x)$ for $x = A, -b/a, B$. On the one hand:

$$\begin{aligned} \varphi_1(A) &= aA + b \\ \varphi_1(-b/a) &= 0 \\ \varphi_1(B) &= 0. \end{aligned}$$

On the other hand:

$$\begin{aligned} \varphi_2(A) &= ReLU(-aA - b) - ReLU(-aA + aA) + \varphi_1(A) \\ &= 0 - 0 + aA + b \\ \varphi_2(-b/a) &= ReLU\big(-a(-b/a) - b \big) - ReLU\big(-a(-b/a) + aA \big) + \varphi_1(A) \\ &= 0 - \varphi_1(A) + \varphi_1(A) = 0 \\ \varphi_2(B) &= ReLU(-aB - b) - ReLU(-aB + aA) + \varphi_1(A) \\ &= (-aB - b) - (-aB + aA) + (aA + b) = 0. \end{aligned}$$

Case 2: $-b/a \geq B$. In this case is is enough to see that $\varphi_1(x) = \varphi_2(x)$ for $x = A, B$. On the one hand:

$$\begin{aligned} \varphi_1(A) &= aA + b \\ \varphi_1(B) &= aB + b. \end{aligned}$$

On the other hand:

$$\begin{aligned} \varphi_2(A) &= ReLU(-aA - b) - ReLU(-aA + aA) + \varphi_1(A) \\ &= 0 - 0 + aA + b \\ \varphi_2(B) &= ReLU(-aB - b) - ReLU(-aB + aA) + \varphi_1(A) \\ &= 0 - (-aB + aA) + (aA + b) = aB + b. \end{aligned}$$

■

In this section, we can assume that all the inner slopes $\{a_j\}$ of any rectified MLP are strictly positive, by the following proposition.

Proposition 5. *Consider the rectified MLP $f_1(x, \theta_1)$ defined on the regression interval $[A, B]$. Then, there exists a rectified MLP $f_2(x; \theta_2)$ with $\theta_2 = \big(a_j, b_j, c_j, d\big)_{j \in \mathcal{J}}$ and $a_j > 0$ for all $j \in \mathcal{J}$, such that $f_2(x, \theta_2) = f_1(x, \theta_1)$ for all $x \in [A, B]$.*

Proof. According Lemma 3, any rectified term in f_1, say $\varphi_1(x) = ReLU(\widetilde{a}_j x + b_j)$ with $\widetilde{a}_j < 0$, can be replaced by the following equivalent function on $[A, B]$:

$$\varphi_2(x) = ReLU(-\widetilde{a}_j x - b_j) - ReLU(-\widetilde{a}_j x + \widetilde{a}_j A) + \varphi_1(A),$$

with strictly positive inner slopes $-\widetilde{a}_j$. Thus, one can obtain f_2 by two steps. Step 1: In f_1, replace any rectified term with negative inner slope, say φ_1, by the corresponding φ_2. Step 2: Rearrange the resulting terms into a rectified MLP, which will have strictly positive inner slopes. ∎

It is well known that the function $ReLU(a_j\, x + b_j)$ is derivable at any point, but at its vertex $-b_j/a_j$. This vertex can be used to define $ReLU(a_j\, x + b_j)$ as follows:

$$ReLU(a_j\, x + b_j) = \begin{cases} 0 & \text{if } x \leq -b_j/a_j \\ a_j\, x + b_j & \text{if } x > -b_j/a_j, \end{cases}$$

where we are assuming $a_j > 0$.

Assumption 5. *For any rectified MLP $f(x; \theta)$ we consider its corresponding set of vertices $V = \{-b_j/a_j\}_{j \in \mathcal{J}}$. Without loss of generality, in this section we assume that V is an ordered set such that*

$$\frac{-b_1}{a_1} < \frac{-b_2}{a_2} < \ldots < \frac{-b_J}{a_J},$$

and that $a_j > 0$ for all $j \in \mathcal{J}$.

In [39] it is shown that, in the case of one-dimensional input, the rectified MLP can potentially learn any continuous function on a compact interval. In order to complement the previous result, Proposition 6 computes the derivative of any rectified MLP. Furthermore, in Corollary 1 it is proved that rectified MLPs have a derivative with moderate variation (in contrast with logistic MLPs). In the experiments of Section 3 we will see that rectified MLPs and the other DC-MLPs, used in nonlinear regression, learn functions whose derivative shows moderate variation, in line with Corollary 1.

Proposition 6. *Let $f(x;\theta)$ be a rectified MLP with $a_j > 0$ for all $j \in \mathcal{J}$ defined on the regression interval $[A, B]$, and V its corresponding set of ordered vertices. If $x_1 \in [A, B] \setminus V$, then*

$$f'(x_1;\theta) = \sum_{j \in \mathcal{J}_1} a_j \, c_j \tag{8}$$

where $\mathcal{J}_1 = \{j \in \mathcal{J} \mid x_1 > -b_j/a_j\}$.

Proof. First, let us recall the derivative of $ReLU(t)$:

$$ReLU'(t) = H(t) = \begin{cases} 0 & \text{if } t < 0 \\ 1 & \text{if } t > 0 \\ \text{undefined} & \text{if } t = 0, \end{cases}$$

where $H(t)$ is the so-called Heaviside function [39].

Second, we partition $\mathcal{J}$ into $\mathcal{J}_1 = \{j \in \mathcal{J} \mid x_1 > -b_j/a_j\}$ and $\mathcal{J}_2 = \{j \in \mathcal{J} \mid x_1 < -b_j/a_j\}$.

Third, let us compute the derivative of $f(x;\theta^*)$ at x_1 :

$$\begin{aligned} f'(x_1;\theta) &= \sum_{j \in \mathcal{J}} c_j \, ReLU'(a_j \, x_1 + b_j) \\ &= \sum_{j \in \mathcal{J}} c_j \, a_j H(a_j \, x_1 + b_j) \\ &= \sum_{j \in \mathcal{J}_1} c_j \, a_j H(a_j \, x_1 + b_j) + \sum_{j \in \mathcal{J}_2} c_j \, a_j H(a_j \, x_1 + b_j) \\ &= \sum_{j \in \mathcal{J}_1} a_j \, c_j. \end{aligned}$$

■

The following corollary shows that rectified MLPs have a derivative which tend to show moderate variation for close points. More specifically, if two points are in the same vertex interval (v_{k-1}, v_k), then their derivatives are equal (statement a)). If two points are in consecutive vertex intervals, then their derivatives, in general, tend to be similar in cases where the products $a_j c_j$ are of similar magnitude for all $j \in \mathcal{J}$ (statement b)).

Corollary 1. *Let $f(x;\theta)$ be a rectified MLP with $a_j > 0$ for all $j \in \mathcal{J}$ defined on the regression interval $[A, B]$, and $V = \{v_k\}_{k \in \mathcal{I}}$ its corresponding set of ordered*

vertices, where $v_k = -b_k/a_k$.

a) If $x_1, x_2 \in (v_{k-1}, v_k)$, *with* $f'(x_1;\theta) \neq 0$ *then* $\dfrac{f'(x_2;\theta)}{f'(x_1;\theta)} = 1.$

b) If $x_1 \in (v_{k-1}, v_k)$ *and*

$x_2 \in (v_k, v_{k+1})$, *with* $f'(x_1;\theta) \neq 0$ *then* $\dfrac{f'(x_2;\theta)}{f'(x_1;\theta)} = 1 + \dfrac{a_k\, c_k}{\sum_{j=1}^{k-1} a_j\, c_j}.$

Proof. a) If $x_1 \in \big(v_{k-1}, v_k\big)$ then, by Assumption 5 we have

$$v_1 < v_2 < \ldots < v_{k-1} < x_1 < v_k < \ldots v_J,$$

and therefore, $\mathcal{J}_1 = \{1, \ldots, k-1\}$. Then by Proposition 6,

$$f'(x_1;\theta) = \sum_{j=1}^{k-1} a_j\, c_j.$$

For the same reason, $f'(x_2;\theta) = \sum_{j=1}^{k-1} a_j\, c_j = f'(x_1;\theta)$.

b) In this case

$$f'(x_2;\theta) = \sum_{j=1}^{k} a_j\, c_j.$$

Thus

$$\frac{f'(x_2;\theta)}{f'(x_1;\theta)} = \frac{\sum_{j=1}^{k} a_j\, c_j}{\sum_{j=1}^{k-1} a_j\, c_j} = 1 + \frac{a_k\, c_k}{\sum_{j=1}^{k-1} a_j\, c_j}.$$

■

3. Numerical results

3.1. Experiments based on synthetic data

As already said, we call SQ-MLP the Multilayer Perceptron with a Squashing activation (logistic, hyperbolic tangent, etc.) and one hidden layer. In this section we compare SQ-MLPs versus DC-MLPs to asses their own capacity to avoid overfitting in one-dimensional nonlinear regression. For this reason we compare

these MLPs without any additional technique to prevent overfitting (L_2 regularization, early stopping, Dropout, etc.). The experiments of this section are based on synthetic data, whereas experiments based on real-world data can be found in Section 3.2. We use PyTorch 1.7.1 [41] to implement our code and conduct our experiments on a processor Intel i7-10750H CPU at 2.60GHz with 16GB of RAM. As we will see, the general conclusion of our experiments is that DC-MLPs can avoid overfitting in contrast to SQ-MLPs. On top of that, in each experiment we study other practical aspects that may affect the overfitting level, namely:

- Experiment 1: Number of data points.
- Experiment 2: Number of neurons.
- Experiment 3: Number of training iterations.
- Experiment 4: Magnitude of the data noise.
- Experiment 5: Probability distribution of the data noise.
- Experiment 6: Squashing versus convex activation functions.

In all of the experiments we consider the following statistical model

$$Y^{(i)} = g(x^{(i)}) + \varepsilon^{(i)} \qquad \varepsilon^{(i)} \sim N(0, \sigma^2), \text{ for all } i \in \mathcal{I},$$

introduced in equation (1) of Section 2. As an exception, in experiment 5 the normally distributed noise is replaced by a uniformly distributed noise. A data set $\mathcal{D} = \{x^{(i)}, y^{(i)}\}_{i \in \mathcal{I}}$, generated by sampling this model, is used to train the NNs, which objective is to approximate the unknown function $g(x)$ as much as possible. Of course, the only information the MLPs know about $g(x)$ is through the data set (with noise) as is the case in real applications.

The parameter values of each experiment are as follows. All the considered MLPs have one hidden layer. These MLPs are trained by using the Mean Squared Error (MSE) loss function combined with the Adam method, based on a constant step length of 0.1. In Table 1 we summarize the default parameter values. The SQ-MLPs and the DC-MLPs are based on the logistic and the softplus activations, respectively in all the experiments but in experiment 6. Notice that in Section 2.4 we have used the rectified activation as the prototypical DC-MLP for simplicity. However, in this section we prefer to use its smooth version, the softplus activation, to obtain smooth regression functions (it is well known that the rectified activation produces piecewise linear graphs as in Fig. 8 (B)).

Table 1: Parameter values

Training				Noise	Data points	Neurons
Loss	Method	Iterations	Step length	σ		(Hidden layer)
MSE	Adam	20,000	0.1	1	200	200

For each experiment we report a figure and a table. On the one hand, each figure has four plots: the two plots on the left column correspond to SQ-MLPs and the two plots on the right column correspond to DC-MLPs, with identical parameters for the two columns, except for the activation function (see, for example, Fig. 3). On the other hand, each table reports the following results obtained after training: the best MSE loss obtained in the training process, two overfitting indexes and the total variation of the derivative of the regression function at the regression points (see, for example, Table 2). The MSE loss corresponds to

$$L(\theta^*) = \frac{1}{I}\sum_{i\in\mathcal{I}} \big(y^{(i)} - f(x^{(i)};\theta^*) \big)^2.$$

An MSE equal to 0 indicates that a global optimizer has been computed. In this case the regression function f has learned the data points, that is, it shows full overfitting. An MSE greater than 0 is preferable since it indicates a lower level of overfitting, if any.

The overfitting index (OI_1) measures the average distance of the trained regression function f to the unknown function g at the data points:

$$OI_1 = \frac{1}{I}\sum_{i\in\mathcal{I}} \mid g(x^{(i)}) - f(x^{(i)};\theta^*) \mid .$$

The closer f and g are, the lower OI_1 is.

The overfitting index (OI_2) measures the percentage of data points for which the trained regression function f disagrees with g by more than ϵ :

$$OI_2 = 100\frac{\#\{\text{Data points such that } \mid g(x^{(i)}) - f(x^{(i)};\theta^*) \mid \geq \epsilon\}}{\#\ \{\text{Data points}\}}\%,$$

where we have set $\epsilon = \sigma/5$ for all the experiments. As before, the closer f and g are, the lower OI_2 is. Notice that OI_1 and OI_2 can only be computed in synthetic

experiments since they require to know function g (normally unknown in real applications).

We also compute the total variation of the derivative of the regression function at the regression points as follows:

$$V_{f'} = \sum_{i \in \mathcal{I}^-} \mid \Delta f'_i \mid = \sum_{i \in \mathcal{I}^-} \mid f'(x_{i+1}; \theta^*) - f'(x_i; \theta^*) \mid,$$

where $\mathcal{I}^- = \{1, \ldots, I-1\}$. For all the regression functions in our experiments, we report the corresponding value of $V_{f'}$.

Summary. As we will see in the following experiments:

- SQ-MLPs show overfitting in all cases, high values of $V_{f'}$ and large derivative values at some points (zigzag-shaped graphs) in line with Propositions 1 and 2.
- In contrast, DC-MLPs show no overfitting in all cases, low values of $V_{f'}$ and low derivative values, in line with Corollary 1.
- The overfitting level of SQ-MLPs results directly proportional to: the number of neurons, the number of training iterations and the magnitude of the data noise. However, the overfitting level decreases as the number of data points increases.

3.1.1. Experiment 1: Does the number of data points affect the overfitting level?

In this experiment we analyze the effect of the number of data points (20 versus 200). The other parameters can be found in Table 1 and the unknown function corresponds to

$$g(x) = 3\ \sin(x/1.5) + (x/3)^2.$$

As the result of the experiment, we observe that the overfitting level decreases as the number of data points increases. See Fig. 3 and Table 2 for details.

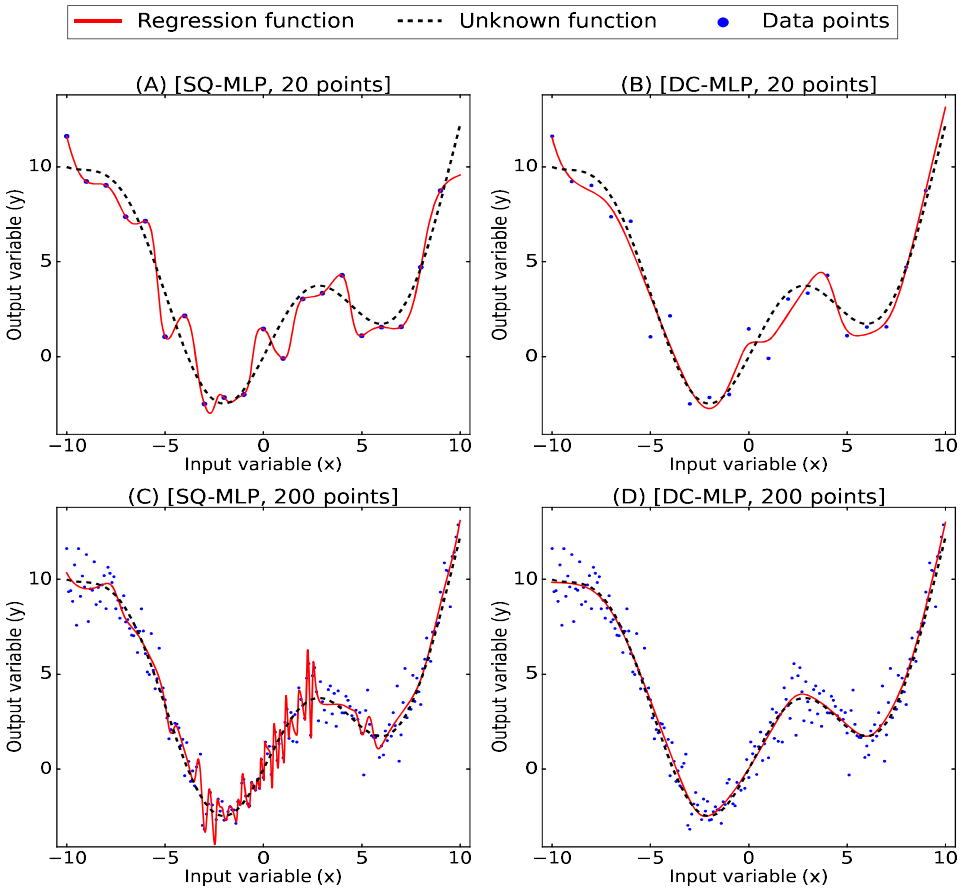

Figure 3: Experiment 1 - Does the number of data points affect the overfitting level? Left column: In SQ-MLPs the overfitting level decreases as the number of data points increases. However, the zigzag becomes steeper as the number of data points increases (compare (A) and (C)). Right column: In DC-MLPs the overfitting level decreases as the number of data points increases. See Table 2 for numerical results.

Table 2: Experiment 1: Numerical results corresponding to Figure 3.

Figure	**Parameters**		**Results obtained after training**			
	MLP	Points	MSE loss	OI_1	OI_2 (%)	$V_{f'}$
1 (A)	SQ	20	0.0000	0.9108	90.00	31.93
1 (B)	DC	20	0.6585	0.5759	75.00	22.85
1 (C)	SQ	200	0.5022	0.4310	74.00	1155.65
1 (D)	DC	200	0.7880	0.1735	33.50	15.57

3.1.2. Experiment 2: Does the number of neurons affect the overfitting level?

In this experiment we analyze the effect of the number of neurons (20 versus 200). The other parameters can be found in Table 1 and we use the same unknown function $g(x)$ as in Experiment 1. After training the corresponding MLPs, we observe that increasing the number of neurons increases the overfitting level of SQ-MLPs, in contrast to DC-MLPs. See Fig. 4 and Table 3 for details.

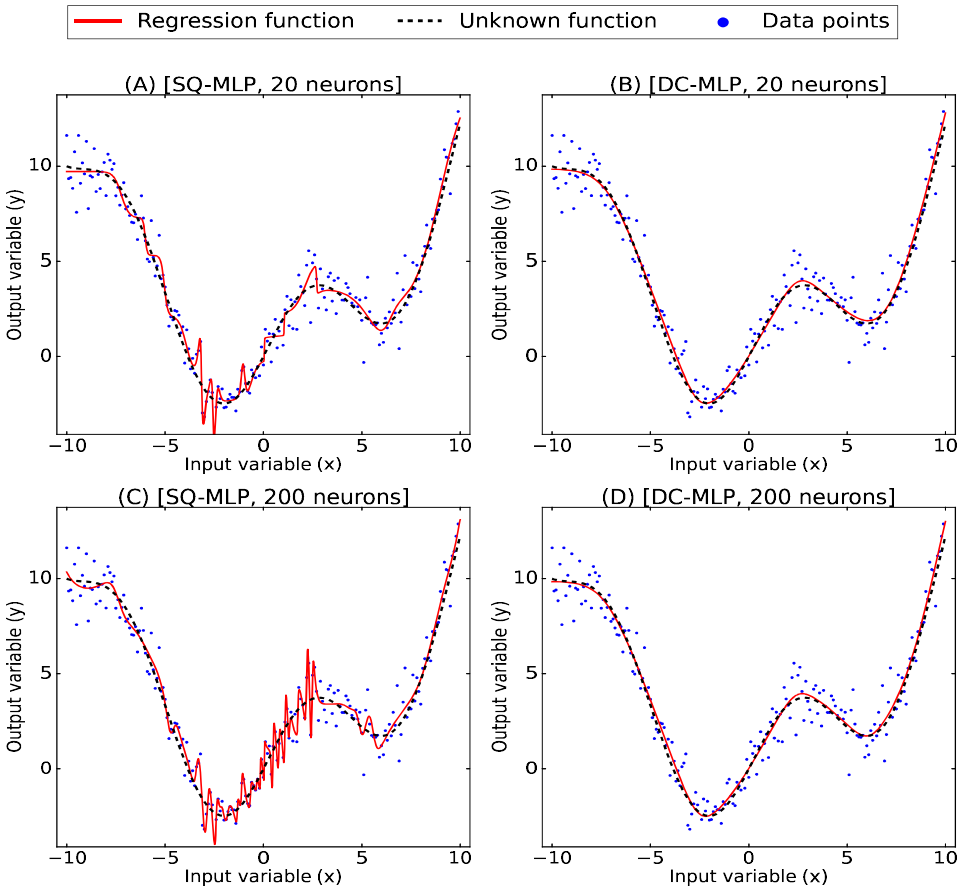

Figure 4: Experiment 2 - Does the number of neurons affect the overfitting level? Left column: The SQ-MLP with 200 neurons shows a higher level of overfitting (the corresponding SQ-MLP can learn more data points). Right column: The DC-MLPs with 20 and 200 neurons show the same level of overfitting. See Table 3 for numerical results.

Table 3: Experiment 2: Numerical results corresponding to Figure 4.

Figure	**Parameters**		**Results obtained after training**			
	MLP	Neurons	MSE loss	OI_1	OI_2 (%)	$V_{f'}$
2 (A)	SQ	20	0.6157	0.3384	59.00	342.83
2 (B)	DC	20	0.7967	0.1691	28.00	15.04
2 (C)	SQ	200	0.5022	0.4310	74.00	1155.65
2 (D)	DC	200	0.7880	0.1735	33.50	15.57

3.1.3. *Experiment 3: Does the number of training iterations affect the overfitting level?*

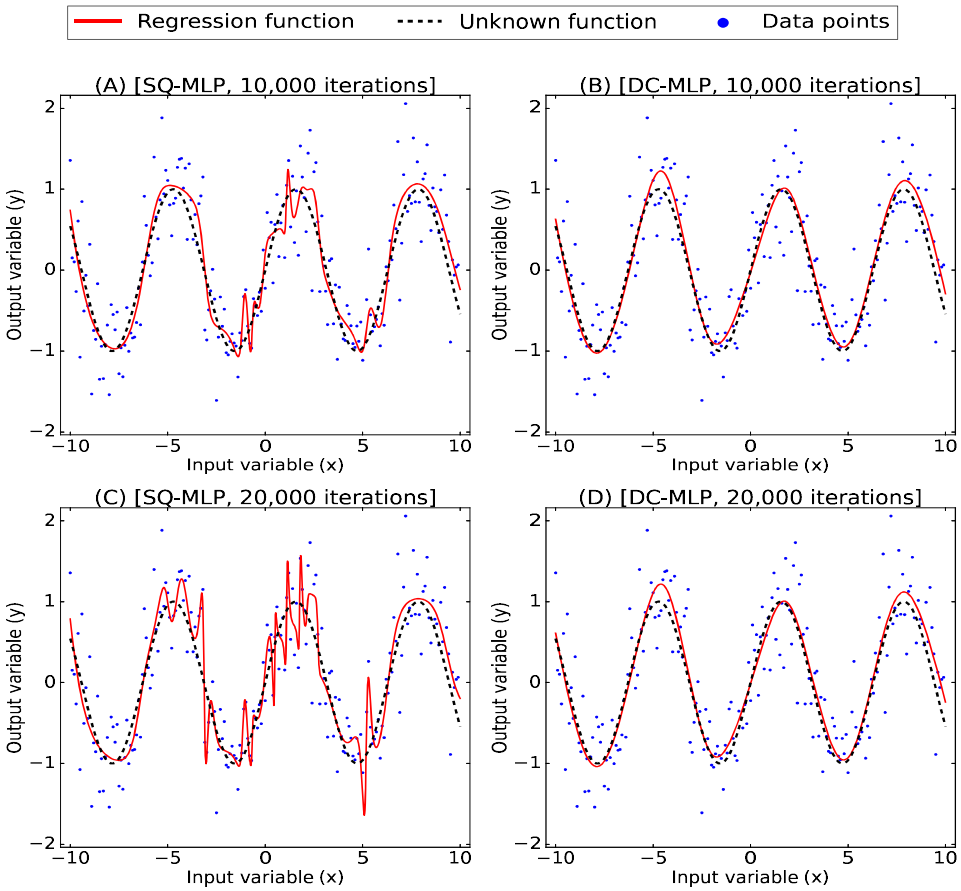

Figure 5: Experiment 3 - Does the number of training iterations affect the overfitting level? Left column: The SQ-MLP trained with 20,000 Adam iterations shows a higher level of overfitting (the corresponding regression function learns more points by further reducing the MSE loss). Right column: The DC-MLPs trained with 10,000 and 20,000 Adam iterations show the same level of overfitting. See Table 4 for numerical results.

Table 4: Experiment 3: Numerical results corresponding to Figure 5.

Figure	**Parameters**		**Results obtained after training**			
	MLP	Iterations	MSE loss	OI_1	OI_2 (%)	$V_{f'}$
3 (A)	SQ	10,000	0.1750	0.1459	61.00	82.21
3 (B)	DC	10,000	0.1994	0.0890	28.00	12.32
3 (C)	SQ	20,000	0.1402	0.1888	65.00	307.33
3 (D)	DC	20,000	0.1992	0.0883	30.50	12.16

In this experiment we analyze the effect of the number of training iterations (10,000 versus 20,000 Adam iterations). The unknown function corresponds to:

$$g(x) = \sin(x).$$

Given that this function takes values in the interval $[-1, 1]$, we set $\sigma = 0.5$ for the normally distributed noise. The other parameters can be found in Table 1. In this experiment, we observe that increasing the number of Adam iterations increases the overfitting level of SQ-MLPs, in contrast to DC-MLPs. See Fig. 5 and Table 4 for details.

3.1.4. *Experiment 4: Does the magnitude of the data noise affect the overfitting level?*

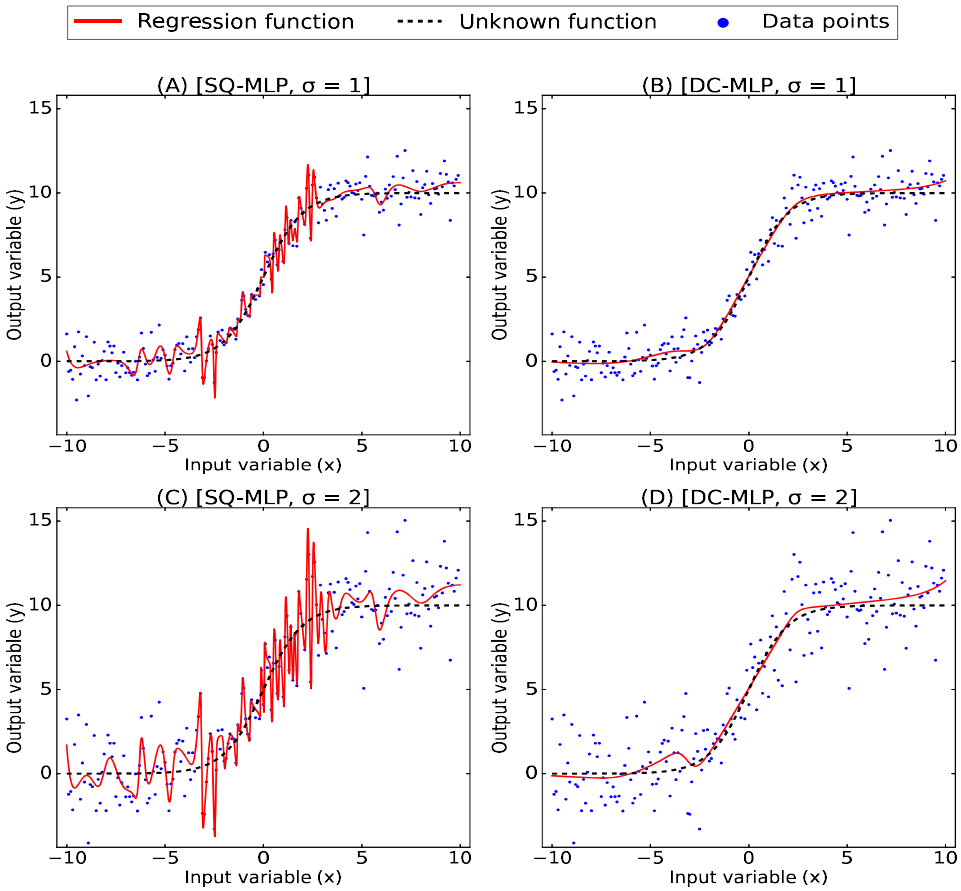

Figure 6: Experiment 4 - Does the magnitude of the data noise affect the overfitting level? Increasing the magnitude of the data noise increases the overfitting level, which is clearly higher for SQ-MLPs (left column) than for DC-MLPs (right column). See Table 5 for numerical results.

Table 5: Experiment 4: Numerical results corresponding to Figure 6.

Figure	**Parameters**		**Results obtained after training**			
	MLP	σ	MSE loss	OI_1	OI_2 (%)	$V_{f'}$
4 (A)	SQ	1	0.4849	0.4289	69.00	1243.91
4 (B)	DC	1	0.7940	0.1681	27.00	5.26
4 (C)	SQ	2	1.7496	0.9525	92.00	2486.31
4 (D)	DC	2	3.1582	0.3345	64.50	8.68

In this experiment we analyze the effect of the magnitude of data noise ($\sigma = 1$ versus $\sigma = 2$). In order to compute OI_2 we have set $\epsilon = 1/5$ for the two values of σ. The other parameters can be found in Table 1 and the unknown function corresponds to:

$$g(x) = 10/(1 + \exp(-x)).$$

Result of the experiment: we observe that increasing the magnitude of the data noise increases the overfitting level, which is clearly higher for SQ-MLPs than for DC-MLPs. See Fig. 6 and Table 5 for details.

3.1.5. Experiment 5: What results would be obtained if the data noise had a uniform distribution?

In this experiment we replace the normal distribution $N(0, \sigma)$ by the uniform distribution $U(-1.5, 1.5)$ to model the data noise ε. Notice that in this case the expectation and standard deviation of ε are 0 and 0.87, respectively. The other parameters can be found in Table 1 and the unknown function corresponds to:

$$g(x) = (x/20)^{-0.75}.$$

Now, for the uniform noise, we obtain results that are similar to those obtained for the normal noise in the previous experiments: a) SQ-MLPs show a higher level of overfitting compared to DC-MLPs. b) Increasing the number of neurons increases the overfitting level of SQ-MLPs, in contrast to DC-MLPs where we observe a slight reduction of the overfitting level. See Fig. 7 and Table 6 for details.

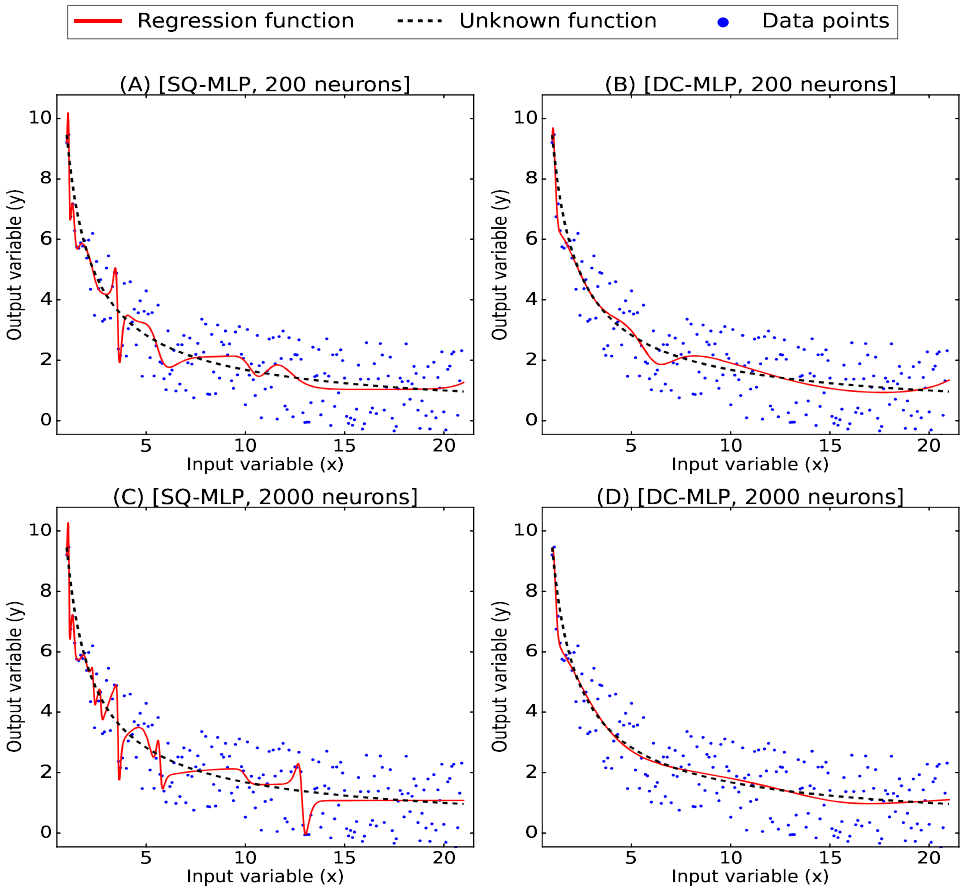

Figure 7: Experiment 5 - What results would be obtained if the data noise had a uniform distribution? Left column: The SQ-MLP with 2000 neurons shows a higher level of overfitting. Right column: The DC-MLP with 2000 neurons shows a slight reduction of the overfitting level. See Table 6 for numerical results.

Table 6: Experiment 5: Numerical results corresponding to Figure 7.

Figure	**Parameters**		**Results obtained after training**			
	MLP	Neurons	MSE loss	OI_1	OI_2 (%)	$V_{f'}$
5 (A)	SQ	200	0.7236	0.2367	52.50	228.87
5 (B)	DC	200	0.7770	0.1754	44.00	56.24
5 (C)	SQ	2000	0.6786	0.2648	46.00	392.08
5 (D)	DC	2000	0.7997	0.1152	11.50	34.15

3.1.6. *Experiment 6: What results would be obtained for other squashing and convex activation functions?*

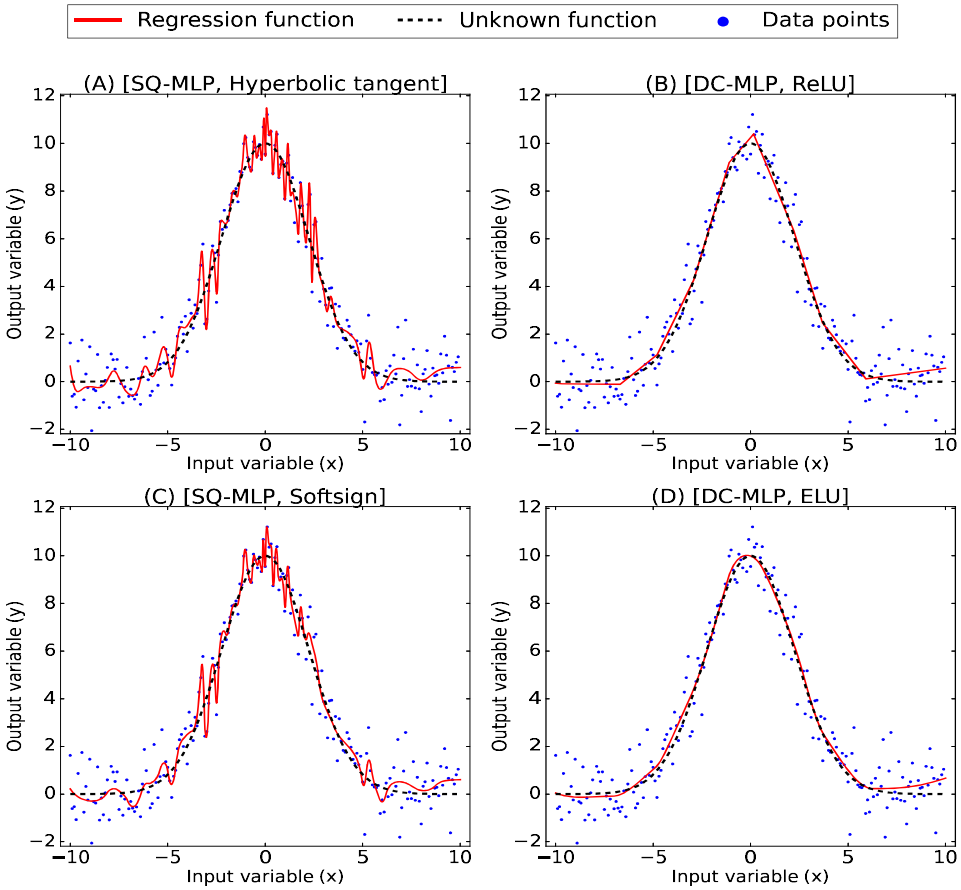

Figure 8: Experiment 6 - What results would be obtained for other squashing and convex activation functions? Left column: SQ-MLPs show a high level of overfitting. Right column: DC-MLPs show virtually no overfitting. See Table 7 for numerical results.

Table 7: Experiment 6: Numerical results corresponding to Figure 8.

Figure	**Parameters**	**Results obtained after training**			
	MLP	MSE loss	OI_1	OI_2 (%)	$V_{f'}$
6 (A)	SQ (Tahn)	0.4586	0.4552	76.00	944.46
6 (B)	DC (ReLU)	0.7672	0.1902	39.50	10.94
6 (C)	SQ (Softsign)	0.5382	0.3798	70.50	499.91
6 (D)	DC (ELU)	0.7801	0.1757	35.00	12.34

So far we have used the logistic activation, which is the prototypical squashing function, and the softplus activation, which is a nonlinear convex function. Now we compare alternative squashing activations, namely hyperbolic tangent and softsign:

$$\tanh(t) = \frac{e^t - e^{-t}}{e^t + e^{-t}}$$
$$\text{softsing}(t) = \frac{t}{1 + \mid t \mid},$$

versus alternative nonlinear convex activations, namely ReLU and ELU:

$$\text{ReLU}(t) = \max\{0, t\}$$
$$\text{ELU}(t) = \begin{cases} \alpha(e^t - 1) & \text{if } t \leq 0 \\ t & \text{if } t > 0, \end{cases}$$

where in the ELU activation, α is a parameter. Notice that squashing and nonlinear convex activations determine the SQ-MLPs and DC-MLPs, respectively. The other parameters can be found in Table 1 and the unknown function corresponds to:

$$g(x) = 10\ \exp(-x^2/10).$$

Now, for the squashing activations, we obtain results that are similar to those obtained for the logistic activation in the previous experiments. For the convex activations, the results are similar to those obtained for the softplus activation. See Fig. 8 and Table 7 for details.

3.2. Experiments based on real-world data

In the experiments of Section 3, we have considered synthetic data, no regularization techniques and shallow MLPs. In this section we compare again SQ-MLPs versus DC-MLPs in the context of one-dimensional nonlinear regression, but in order to enhance the previous numerical results, we consider real-wold data, regularization techniques and deep MLPs. Hardware and software settings are identical to that of Section 3. Now, we consider the following ten data sets from the UCI machine learning repository [42], which contains real-wold data:

1. Bike Sharing.
2. Istanbul Stock Exchange.
3. Parking Birmingham.
4. Auto MPG.
5. Productivity Prediction of Garment Employees.
6. Air Quality.
7. Beijing PM2.5 Data.
8. AI4I 2020 Predictive Maintenance Data Set.
9. EEG Steady-State Visual Evoked Potential Signals.
10. SML2010.

As we saw in Definition 1, shallow MLPs have only one hidden layer. In a preliminary computational test, we have compared shallow SQ-MLPs versus shallow DC-MLPs by using the above ten data sets and have obtained similar results (slightly better for DC-MLPs regarding the overfitting level). On the other hand, MLPs with two or more hidden layers are usually termed deep MLPs. The rest of this section is dedicated to compare deep SQ-MLPs versus deep DC-MLPs. Analogously to Definition 3, given an MLP $f(x,\theta)$, we say that it is a deep DC-MLP if it is deep and can be decomposed as follows $f(x,\theta) = f_1(x,\theta_1)+f_2(x,\theta_2)$, where $\theta = (\theta_1,\theta_2)$, f_1 is a convex function in x and f_2 is a concave function in x (see [19]). It is not difficult to obtain deep DC-MLPs by the following proposition, analogous to Proposition 4.

Proposition 7 ([19]). *If $f(x;\theta)$ is a deep MLP with: a) Hidden layers indexed by $\ell = 1, \cdots, L-1$. b) A nonlinear activation function which is increasing convex (softplus, ReLU, leaky ReLU, ELU, etc.) in all the hidden layers. c) The identity activation function in the output layer L. d) Non-negative weights $w_{ij}^{(\ell)} \geq 0$ in the hidden layers $\ell = 2, \cdots, L-1$. Then it is a deep DC-MLP (notice that bias $b_i^{(\ell)}$ have no sign constraints for all ℓ and weights $w_{ij}^{(\ell)}$ have no sign constraints for $\ell = 1, L$).*

In practice all we need to do to fulfill this proposition to obtain a deep DC-MLP is: 1) Take a deep MLP with a nonlinear activation function, which is increasing convex, in all the layers but the output one, which has the identity activation. 2) At the end of each iteration of the training method (SGD, Adam, etc.), set equal to 0 all the weights with a negative value, in layers $\ell = 2, \cdots, L-1$.

The training parameters for each experiment are as follows. The SQ-MLPs and the DC-MLPs are based on the logistic and the softplus activations, respectively in all the experiments. All the considered MLPs have two hidden layers ($L = 3$), with 200 neurons each. These MLPs are trained by using the Mean Squared Error (MSE) loss function combined with the Adam optimization method, based on a constant step length equal to 0.01, for data sets 1 to 5, and 0.001, for data sets 6 to 10.

For each approach, we have tuned the corresponding regularization parameters independently. On the one hand, for the SQ approach, we have used the following parameter values in all the experiments: $\lambda = 5 \cdot 10^{-5}$ for the L_2 penalty term $\lambda \|w\|_2^2$, and early stop patience equal to 100 iterations (number of iterations with no loss improvement with the validation set, after which training will be stopped). On the other hand, for the DC approach, we have used the following parameter values in all the experiment: $\lambda = 5 \cdot 10^{-6}$ and early stop patience equal to 500 iterations.

Furthermore, we have applied Dropout on data sets 1 to 5 and DropConnect on data sets 6 to 10. Dropout has been applied in layers $\ell = 1, 2$, with Dropout rate $p = 0.75$ and $p = 0.30$ for the SQ and DC approaches, respectively (in this case, p is the probability of setting a given activation in a layer to zero). On the other hand, DropConnect has been applied in layers $\ell = 2, 3$, with DropConnect rate $p = 0.30$ for the SQ and DC approaches (in this case, p is the probability of setting a given weight in a layer to zero) [43].

For each experiment we report a figure which has four plots labeled from (A) to (D). In each plot we represent the two nonlinear regression functions corresponding to the SQ and DC approaches. Notice that the plots of the ten data sets

have been scaled to the square $[0,\ 1] \times [0,\ 1]$ for a homogeneous presentation. Plot (A) is obtained without any regularization technique. In contrast, plots (B), (C) and (D) are obtained by using regularization L_2, early stopping and Dropout/-DropConnect, respectively for the SQ and DC approaches (see, for example, Fig. 9).

Summary. Next, we summarize the general trends for plots (A) to (D) that can be observed in the following ten experiments (see Fig. 9 to Fig. 17):

- Plot (A) - MLPs trained without regularization: Approach SQ shows overfitting in all the cases, but for data set 3. Approach DC shows no overfitting in all the cases.
- Plot (B) - MLPs trained with L_2 regularization: The results of both approaches are almost the same.
- Plot (C) - MLPs trained with early stopping regularization: The results of both approaches are similar.
- Plot (D) - MLPs trained with Dropout/DropConnect regularization (we have used Dropout in data sets 1 to 5 and DropConnect in data sets 6 to 10): The results of both approaches are also similar, but for data set 1 where the SQ result seems unsatisfactory for practical purposes.
- Therefore, we can conclude that the DC approach self-regularizes in contrast to the SQ approach, which requires regularization and obtains the best results by using L_2 regularization.

3.2.1. Data set 1: Bike Sharing

This data set contains hourly and daily counts of rental bicycles throughout 2011 and 2012 on the Capital bikeshare system with corresponding weather and seasonal information in Washington, D.C., USA [44]. To set our nonlinear regression problem we consider the following variables from the data set:

$x^{(i)} =$ 'Day i-th of the year (starting from Jan 1, 2012)'

$y^{(i)} =$ 'Count of total rental bikes (per day) including both casual and registered' (varible 'cnt' in the data set)

$i \in \mathcal{I} = \{1, \ldots, 366\}$.

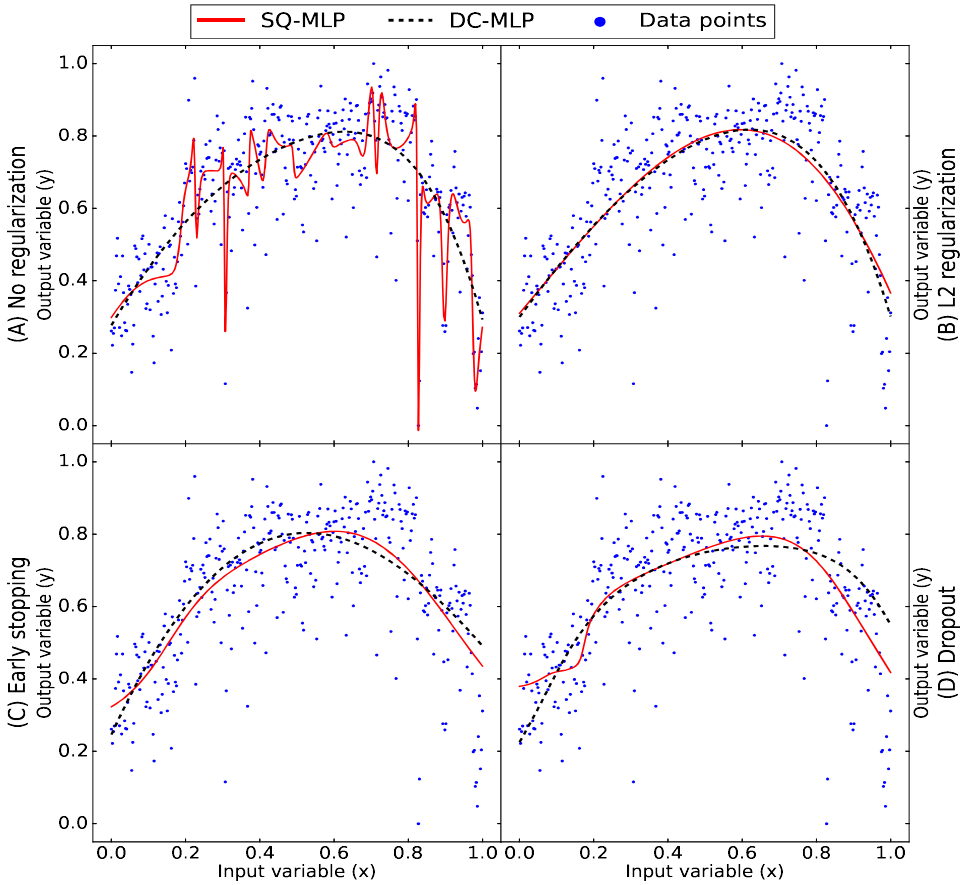

Figure 9: Data set 1 - Bike Sharing. The regression function estimates the total rental bikes per day, along the whole year. Notice that the plots of the five data sets have been scaled to the square $[0,\ 1] \times [0,\ 1]$ for a homogeneous presentation. Plot (A) - MLPs trained without regularization: Approach SQ shows overfitting, in sharp contrast with approach DC. Plot (B) - MLPs trained with L_2 regularization: The results of both approaches are almost the same. Plot (C) - MLPs trained with early stopping regularization: The results of both approaches are similar. Plot (D) - MLPs trained with Dropout regularization: Overfitting is avoided by both approaches, but the SQ regression function seems unsatisfactory for practical purposes.

Table 8: Data set 1 - Bike Sharing: Numerical results corresponding to Figure 9.

Figure	**Parameters**		**Results obtained after training**	
	MLP	Regularization	MSE loss	$V_{f'}$
8 (A)	SQ	-	0.0091	1904.98
8 (A)	DC	-	0.0177	5.30
8 (B)	SQ	L_2	0.0209	3.49
8 (B)	DC	L_2	0.0182	4.79
8 (C)	SQ	Early stopping	0.0167	4.10
8 (C)	DC	Early stopping	0.0219	3.41
8 (D)	SQ	Dropout	0.0202	11.87
8 (D)	DC	Dropout	0.0198	4.01

3.2.2. Data set 2: Istanbul Stock Exchange

This data set includes the returns of the Istanbul Stock Exchange (ISE) with seven other international indices: SP, DAX, FTSE, NIKKEI, BOVESPA, MSCE-EU, MSCI-EM from January 5, 2009 to February 22, 2011 [45]. To set our non-linear regression problem we consider the following variables from the data set:

$x^{(i)} =$ 'Day i-th of the year (starting from Jan 5, 2009)'

$y^{(i)} =$ 'Cumulative return of ISE at the end of the i-th day, from Jan 5, 2009' (computed from varible 'TL BASED ISE' in the dataset)

$i \in \mathcal{I} = \{1, \ldots, 536\}$.

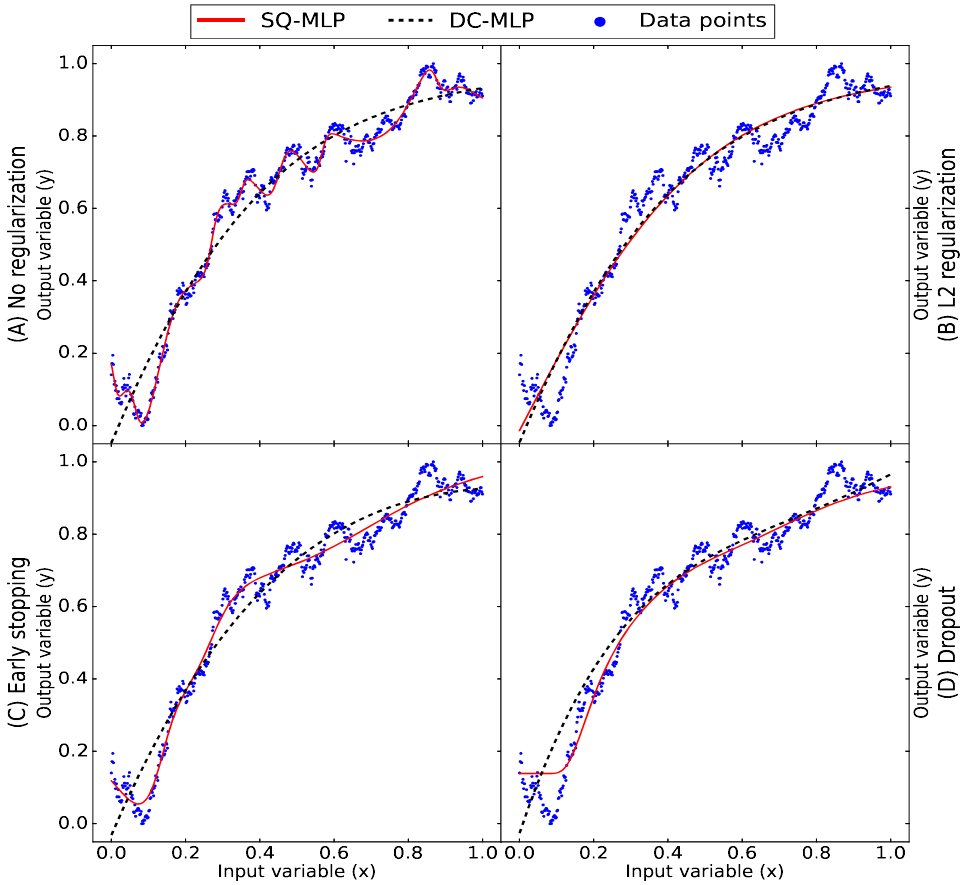

Figure 10: Data set 2 - Istanbul Stock Exchange (ISE). The regression function approximates the daily cumulative return of the ISE, along two years. The corresponding comments for plots (A) to (D) are similar to those in Fig. 9, except that Dropout now obtains a SQ regression function that seems acceptable for practical purposes.

Table 9: Data set 2 - Istanbul Stock Exchange: Numerical results corresponding to Figure 10.

Figure	**Parameters**		**Results obtained after training**	
	MLP	Regularization	MSE loss	$V_{f'}$
9 (A)	SQ	-	0.0004	74.65
9 (A)	DC	-	0.0035	2.30
9 (B)	SQ	L_2	0.0052	1.84
9 (B)	DC	L_2	0.0037	2.27
9 (C)	SQ	Early stopping	0.0053	10.01
9 (C)	DC	Early stopping	0.0038	2.29
9 (D)	SQ	Dropout	0.0076	5.73
9 (D)	DC	Dropout	0.0063	2.72

3.2.3. Data set 3: Parking Birmingham

This data set contains data collected from Birmingham parking lots managed by Birmingham City Council's NCP, UK Open Government License (OGL) [46]. To set our nonlinear regression problem we consider the occupancy in the parking lot with CodeNumber BHMBCCMKT01 during the 4 Saturdays of November, 2016. The occupancy is measured at 18 time steps every Saturday (from 8:00 am till 16:30, every 30 minutes approximately). We consider the following variables from the data set:

$x^{(ij)} =$ 'i-th time step corresponding to the j-th Saturday'

$y^{(ij)} =$ 'Number of cars in the parking lot corresponding to $x^{(ij)}$'
(variable 'Occupancy' in the data set)
$i \in \mathcal{I} = \{1, \ldots, 18\},\ j \in \mathcal{J} = \{1, \ldots, 4\}.$

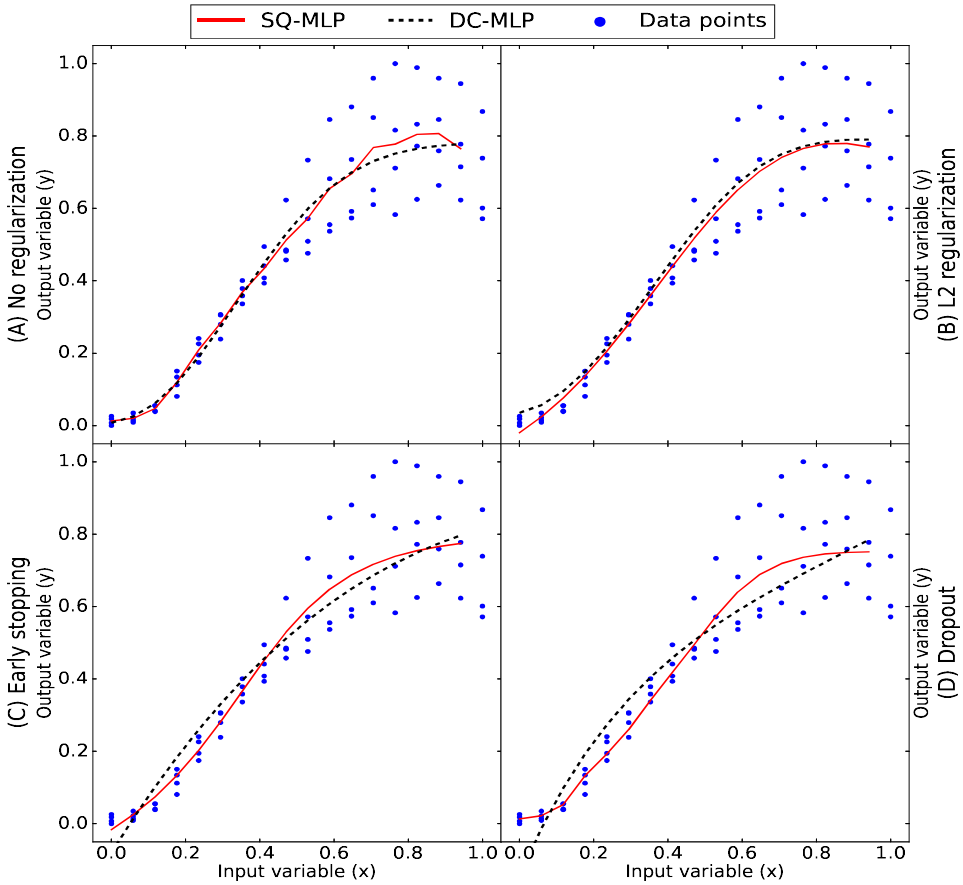

Figure 11: Data set 3 - Parking Birmingham. The regression function estimates the number of cars in a parking lot, for Saturdays in November. The corresponding comments for plots (A) to (D) are similar to those in Fig. 10, except that now the SQ regression function does not show overfitting in plot (A).

Table 10: Data set 3 - Parking Birmingham: Numerical results corresponding to Figure 11.

Figure	**Parameters**		**Results obtained after training**	
	MLP	Regularization	MSE loss	$V_{f'}$
10 (A)	SQ	-	0.0083	32.87
10 (A)	DC	-	0.0091	11.72
10 (B)	SQ	L_2	0.0111	12.83
10 (B)	DC	L_2	0.0094	11.51
10 (C)	SQ	Early stopping	0.0088	10.44
10 (C)	DC	Early stopping	0.0116	9.36
10 (D)	SQ	Dropout	0.0152	14.00
10 (D)	DC	Dropout	0.0167	13.42

3.2.4. Data set 4: Auto MPG

The data concerns city-cycle fuel consumption in miles per gallon, to be predicted in terms of 3 multivalued discrete and 5 continuous attributes [47]. To set our nonlinear regression problem we consider the following variables from the data set:

$$
\begin{aligned}
x^{(i)} &= \text{‘Horsepower (hp)’} \\
y^{(i)} &= \text{‘Miles per gallon’} \\
&\quad \text{(variables ‘horsepower’ and ‘mpg’ in the data set)} \\
&\quad i \in \mathcal{I} = \{1, \ldots, 392\}.
\end{aligned}
$$

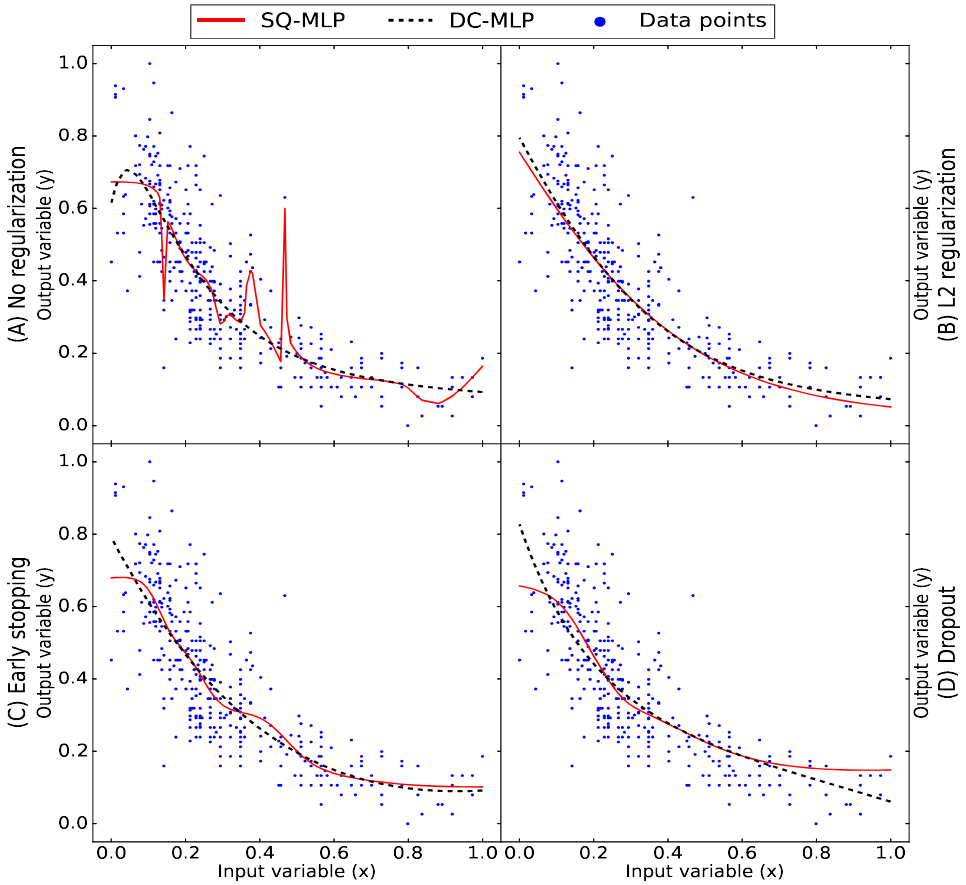

Figure 12: Data set 4 - Auto MPG. The regression function estimates the miles per gallon as a function of the car horsepower. In plots (A), (C) and (D), the corresponding DC regression function seems more appropriate than the SQ counterpart, for practical purposes. In plot (B) both regression functions are similar.

Table 11: Data set 4 - Auto MPG: Numerical results corresponding to Figure 12.

Figure	**Parameters**		**Results obtained after training**	
	MLP	Regularization	MSE loss	$V_{f'}$
11 (A)	SQ	-	0.0108	329.00
11 (A)	DC	-	0.0129	8.19
11 (B)	SQ	L_2	0.0152	1.51
11 (B)	DC	L_2	0.0136	1.84
11 (C)	SQ	Early stopping	0.0133	7.41
11 (C)	DC	Early stopping	0.0141	2.01
11 (D)	SQ	Dropout	0.0145	3.47
11 (D)	DC	Dropout	0.0158	2.75

3.2.5. Data set 5: Productivity Prediction of Garment Employees

This data set includes important attributes of the garment manufacturing process and employee productivity, which have been collected manually and also validated by industry experts [48]. To set our nonlinear regression problem we consider the following variables from the data set:

$x^{(i)} =$ 'Financial incentive (in Bangladeshi takas)'

$y^{(i)} =$ 'Productivity (in the range [0, 1])'

(variables 'incentive' and 'productivity' in the data set)

$i \in \mathcal{I} = \{1, \ldots, 582\}$.

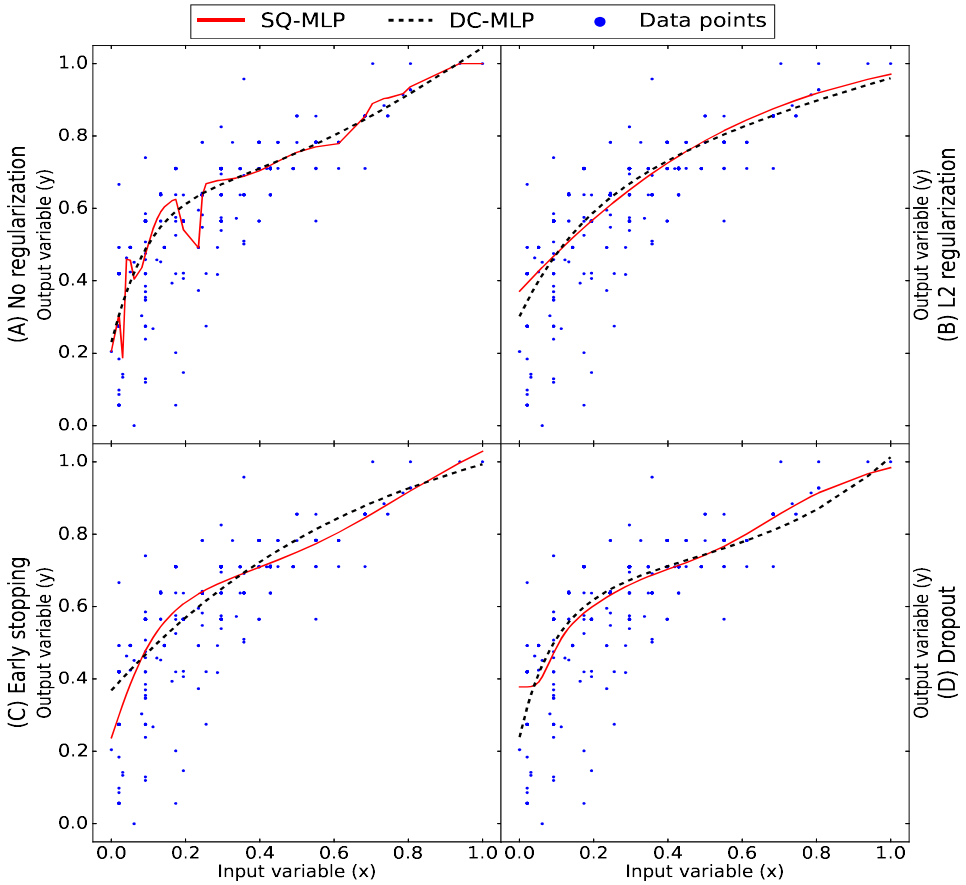

Figure 13: Data set 5 - Productivity Prediction of Garment Employees. The regression function estimates the productivity as a function of the financial incentive. Plot (A) - MLPs trained without regularization: Approach SQ shows overfitting, in contrast with approach DC. Plot (B) - MLPs trained with L_2 regularization: The results of both approaches are almost the same. Plot (C) - MLPs trained with early stopping regularization: The results of both approaches are similar. Plot (D) - MLPs trained with Dropout regularization: The results of both approaches are similar.

Table 12: Data set 5 - Productivity Prediction of Garment Employees: Numerical results corresponding to Figure 13.

Figure	**Parameters**		**Results obtained after training**	
MLP	MLP	Regularization	MSE loss	$V_{f'}$
12 (A)	SQ	-	0.0077	260.81
12 (A)	DC	-	0.0085	3.77
12 (B)	SQ	L_2	0.0110	0.89
12 (B)	DC	L_2	0.0094	1.83
12 (C)	SQ	Early stopping	0.0088	3.00
12 (C)	DC	Early stopping	0.0104	0.85
12 (D)	SQ	Dropout	0.0108	4.80
12 (D)	DC	Dropout	0.0092	4.76

3.2.6. Data set 6: Air Quality

This data set contains the responses of a gas multisensor device located on a field within an Italian city. Hourly responses averages are recorded along with gas concentrations references from a certified analyzer [49]. To set our nonlinear regression problem we consider the hourly averaged NO_x sensor response from March 11, 2004 till March 21, 2004:

$x^{(i)} =$ 'Hourly i-th time step (starting from March 11, 2004 till March 21, 2004)'

$y^{(i)} =$ 'Average NO_x sensor response corresponding to $x^{(i)}$ ($\mu g/m^3$)'
(variables 'Date', 'Time', and 'PT08.S3(NOx)' in the data set)
$i \in \mathcal{I} = \{1, \ldots, 264\}$.

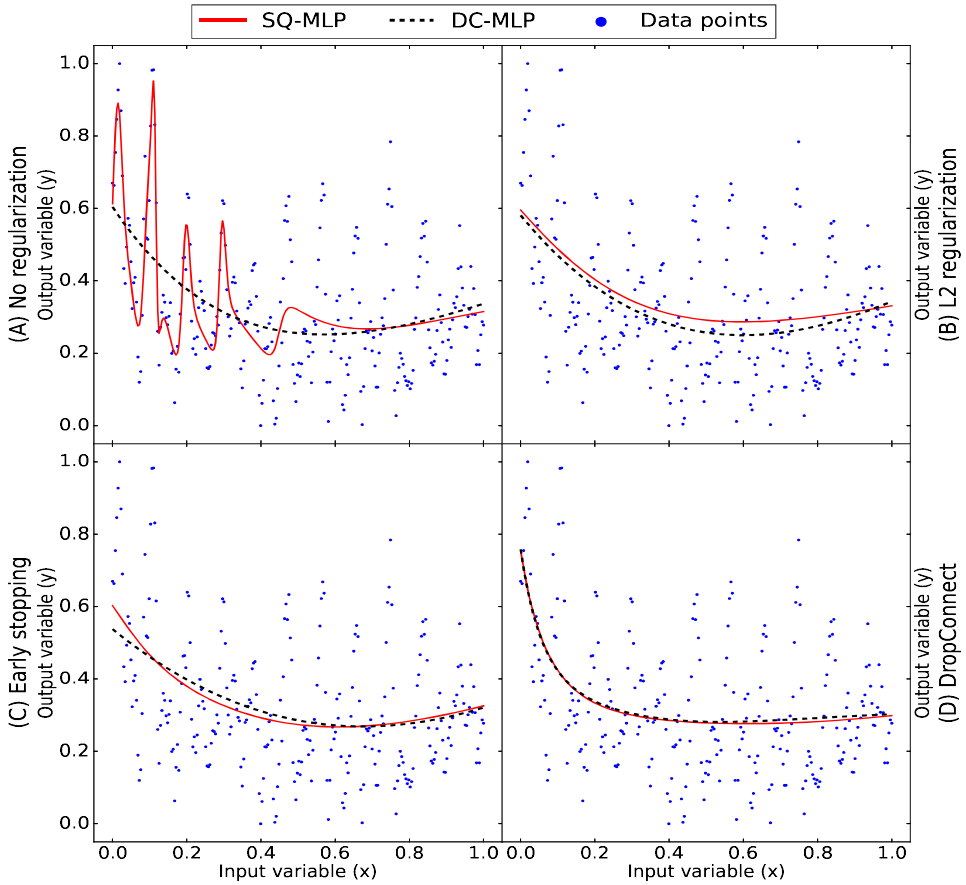

Figure 14: Data set 6 - Air Quality. The regression function estimates the hourly averaged NO_x sensor response, along 11 days. In data sets 6 to 10 we use DropConnect instead of Dropout (see plot (D)).

Table 13: Data set 6 - Air Quality: Numerical results corresponding to Figure 3.2.6.

Figure	**Parameters**		**Results obtained after training**	
MLP	MLP	Regularization	MSE loss	$V_{f'}$
13 (A)	SQ	-	0.0180	492.97
13 (A)	DC	-	0.0290	1.85
13 (B)	SQ	L_2	0.0323	1.34
13 (B)	DC	L_2	0.0297	1.68
13 (C)	SQ	Early stopping	0.0276	1.76
13 (C)	DC	Early stopping	0.0282	1.15
13 (D)	SQ	DropConnect	0.0280	6.65
13 (D)	DC	DropConnect	0.0280	7.18

3.2.7. Data set 7: Beijing PM2.5 Data

This data set contains the $PM_{2.5}$ data of the US Embassy in Beijing [50]. $PM_{2.5}$ stands for Particulate Matter 2.5, fine airborne particles with a diameter of 2.5 micrometers or smaller. To set our nonlinear regression problem we consider the hourly dew point temperature from January 10, 2010 until January 19, 2010:

$x^{(i)} =$ 'Hourly i-th time step (starting from January 10, 2010 until January 19, 2010)'

$y^{(i)} =$ 'Dew point temperature corresponding to $x^{(i)}$ (°C)' (variables 'year', 'month', 'day', 'hour', and 'DEWP' in the data set) $i \in \mathcal{I} = \{1, \ldots, 240\}$.

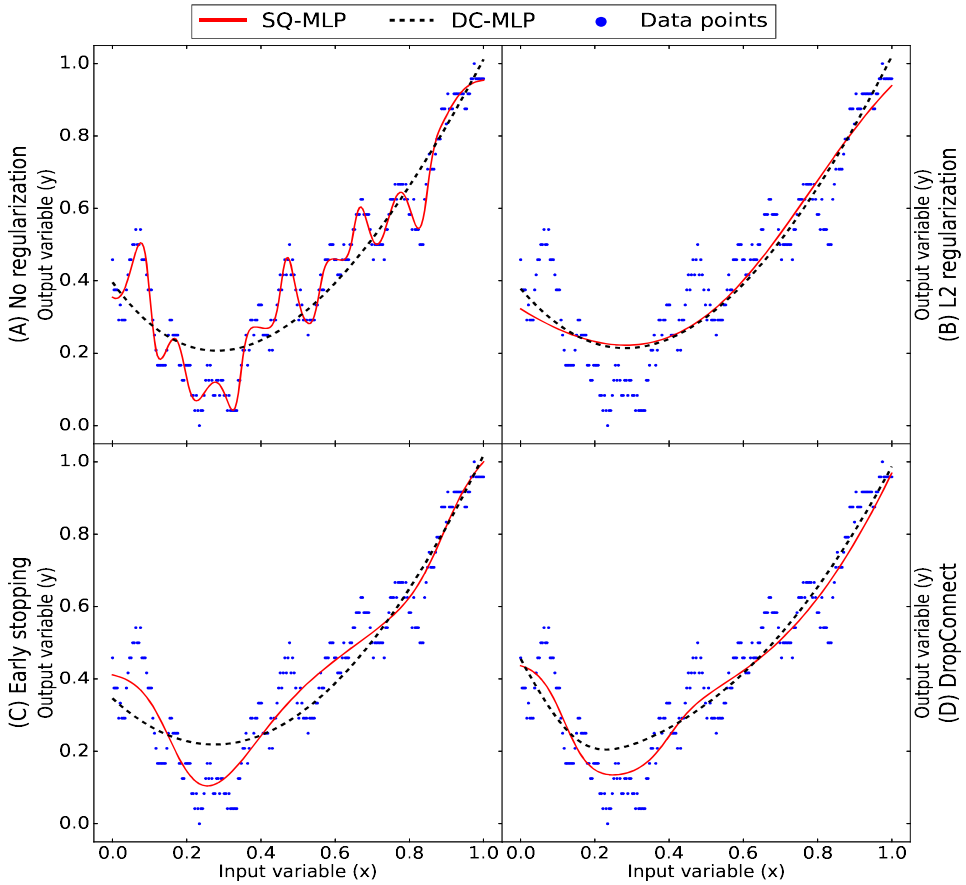

Figure 15: Data set 7 - Beijing PM2.5 Data. The regression function estimates the hourly dew point temperature, along 10 days. The SQ regression functions clearly depend on the regularization method, in contrast to the DC regression functions.

Table 14: Data set 7 - Beijing PM2.5 Data: Numerical results corresponding to Figure 3.2.7.

Figure	**Parameters**		**Results obtained after training**	
	MLP	Regularization	MSE loss	$V_{f'}$
14 (A)	SQ	-	0.0011	154.61
14 (A)	DC	-	0.0084	3.34
14 (B)	SQ	L_2	0.0129	2.54
14 (B)	DC	L_2	0.0094	3.23
14 (C)	SQ	Early stopping	0.0044	8.84
14 (C)	DC	Early stopping	0.0089	3.03
14 (D)	SQ	DropConnect	0.0076	7.99
14 (D)	DC	DropConnect	0.0082	3.77

3.2.8. Data set 8: AI4I 2020 Predictive Maintenance Data Set

The AI4I 2020 Predictive Maintenance Data set is a synthetic data set that reflects real predictive maintenance data encountered in industry [51]. This data set was introduced in the Artificial Intelligence for Industries conference (AI4I) [52]. To set our nonlinear regression problem we consider the air temperature generated by a random walk process:

$x^{(i)} =$ 'i-th time step of the random walk process'
$y^{(i)} =$ 'Air temperature corresponding to $x^{(i)}$ (K)'
(variable 'UDI' and 'Air temperature [K]' in the data set)
$i \in \mathcal{I} = \{1, \ldots, 10000\}$.

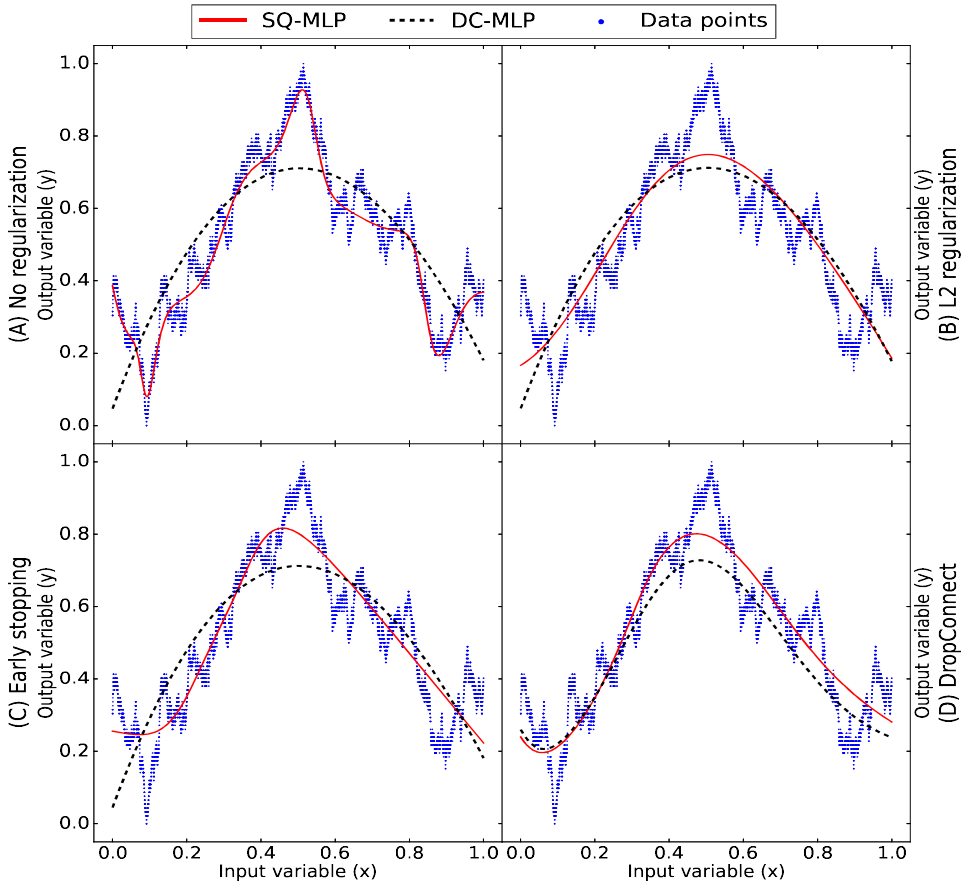

Figure 16: Data set 8 - AI4I 2020 Predictive Maintenance Data set. The regression function estimates the temperature, along 10000 time steps.

Table 15: Data set 8 - AI4I 2020 Predictive Maintenance Data set: Numerical results corresponding to Figure 3.2.8.

Figure	**Parameters**		**Results obtained after training**	
	MLP	Regularization	MSE loss	$V_{f'}$
15 (A)	SQ	-	0.0024	65.55
15 (A)	DC	-	0.0142	4.76
15 (B)	SQ	L_2	0.0140	4.48
15 (B)	DC	L_2	0.0149	4.77
15 (C)	SQ	Early stopping	0.0077	6.23
15 (C)	DC	Early stopping	0.0142	4.74
15 (D)	SQP	DropConnect	0.0080	8.80
15 (D)	DC	DropConnect	0.0088	7.98

3.2.9. Data set 9: EEG Steady-State Visual Evoked Potential Signals

This database consists on 30 subjects performing Brain Computer Interface for Steady State Visual Evoked Potentials (BCI-SSVEP) [53]. To set our nonlinear regression problem we consider the brain signals for the experiment conditions A001SB1_1 (first experiment on the Five Box Visual Test 1 for the first subject of the group A). Specifically, we analyze the values recorded by the left occipital lobe (O1) electrode from step 820 to step 1640. We consider the following variables from the data set:

$$
\begin{aligned}
x^{(i)} &= \text{‘i-th time step (starting from step 820 until step 1640)’} \\
y^{(i)} &= \text{‘O1 electrode reading corresponding to } x^{(i)} \text{ (µV)’} \\
&\quad \text{(variable ‘O1’ in the data set)} \\
&\quad i \in \mathcal{I} = \{1, \ldots, 820\}.
\end{aligned}
$$

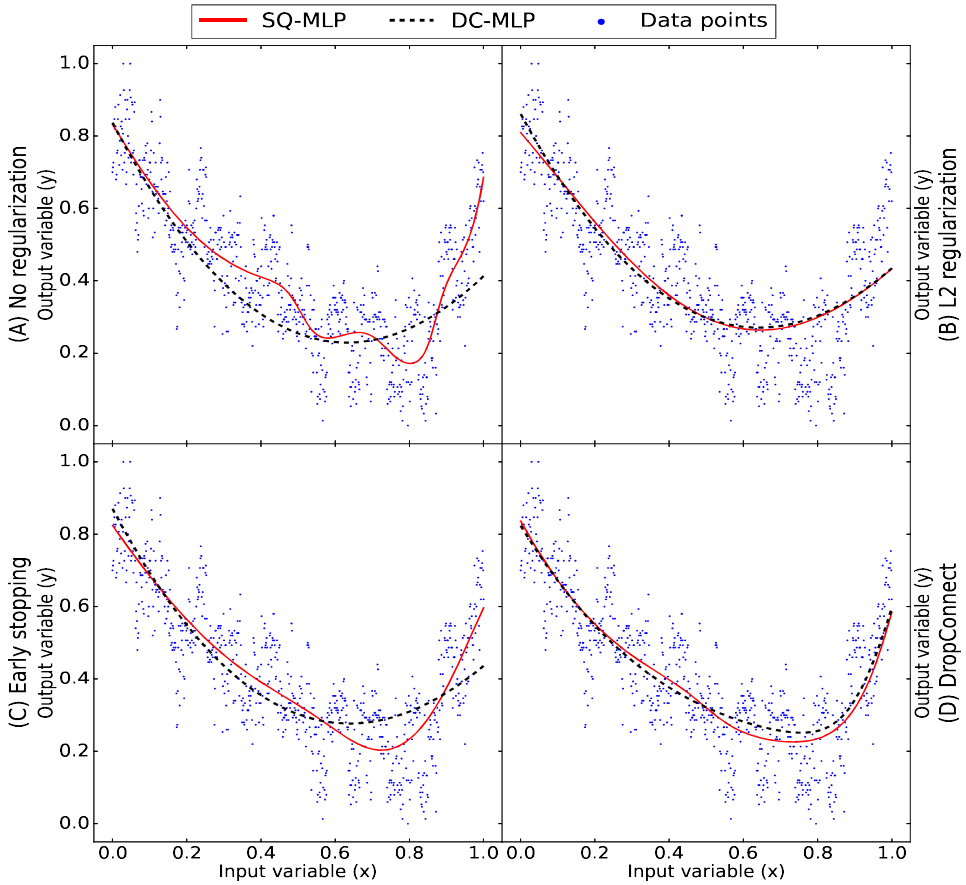

Figure 17: Data set 9 - EEG Steady-State Visual Evoked Potential Signals. The regression function estimates the brain signal of the left occipital lobe electrode, along 820 time steps.

Table 16: Data set 9 - EEG Steady-State Visual Evoked Potential Signals: Numerical results corresponding to Figure 3.2.9.

Figure	**Parameters**		**Results obtained after training**	
	MLP	Regularization	MSE loss	$V_{f'}$
16 (A)	SQ	–	0.0096	17.29
16 (A)	DC	–	0.0141	2.94
16 (B)	SQ	L_2	0.0169	2.18
16 (B)	DC	L_2	0.0143	2.78
16 (C)	SQ	Early stopping	0.0102	3.98
16 (C)	DC	Early stopping	0.0135	2.79
16 (D)	SQ	DropConnect	0.0133	6.09
16 (D)	DC	DropConnect	0.0128	4.96

3.2.10. Data set 10: SML2010

This data set is collected from a monitor system mounted in a domotic house, specifically a solar-powered house known as Small Medium Large System (SML-system) [54]. It corresponds to approximately 40 days of monitoring data [55]. To set our nonlinear regression problem we consider the outdoor relative humidity (%) from March 19, 2012 until March 20, 2012, measured every 15 minutes throughout both days:

$x^{(i)} =$ 'i-th time step (starting from March 19, 2012 until March 20, 2012)'

$y^{(i)} =$ 'Outdoor relative humidity corresponding to $x^{(i)}$ (%)'
(variables '1:Date', '2:Time', and '23:Humidity_Outside_Sensor' in the data set)
$i \in \mathcal{I} = \{1, \ldots, 192\}$.

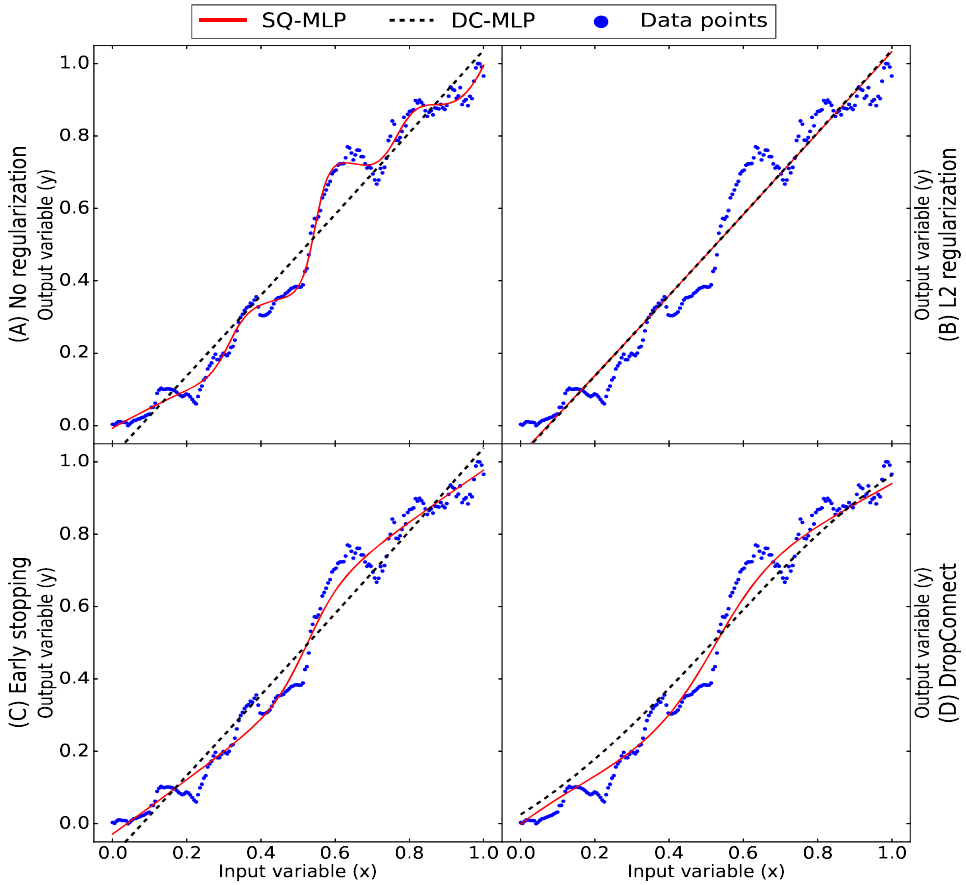

Figure 18: Data set 10 - SML2010. The regression function estimates the outdoor relative humidity (%), along two days.

Table 17: Data set 10 - SML2010: Numerical results corresponding to Figure 3.2.10

Figure	**Parameters**		**Results obtained after training**	
	MLP	Regularization	MSE loss	$V_{f'}$
17 (A)	SQ	-	0.0005	19.46
17 (A)	DC	-	0.0037	0.04
17 (B)	SQ	L_2	0.0049	0.06
17 (B)	DC	L_2	0.0039	0.04
17 (C)	SQ	Early stopping	0.0014	2.94
17 (C)	DC	Early stopping	0.0039	0.05
17 (D)	SQ	DropConnect	0.0051	2.51
17 (D)	DC	DropConnect	0.0037	0.87

4. Conclusions

In this work, we have shown the ability of the Difference of Convex Multilayer Perceptron (DC-MLP) to avoid overfitting in one-dimensional nonlinear regression. That is, DC-MLPs self-regularize. It has been shown that shallow MLPs with a convex activation (ReLU, softplus, etc.) fall in the class of DC-MLPs. On the other hand, we have called SQ-MLP the shallow MLP with a Squashing activation (logistic, hyperbolic tangent, etc.). In the numerical experiments, we have shown that DC-MLPs used for one-dimensional nonlinear regression avoid overfitting, in contrast with SQ-MLPs.

In Section 3.1, we have carried out a set of six nonlinear regression experiments based on synthetic data. In these experiments, no regularization techniques have been considered – as L_2-regularization, early stopping, Dropout, etc. – enabling the detection of self-regularization, i.e., avoid overfitting. In this case, DC-MLPs have shown virtually no overfitting, whereas SQ-MLPs have overfitted in all cases. Furthermore, the overfitting level of SQ-MLPs has resulted directly proportional to: the number of neurons, the number of training iterations and the magnitude of the data noise. However, it has resulted inversely proportional to the number of data points.

Additionally, in Section 3.2, we have carried out a set of ten nonlinear regression experiments based on real-world data. In these experiments, SQ-MLPs have

required to be complemented with some regularization technique to avoid overfitting, in contrast with DC-MLPs. That is, DC-MLPs have shown an intrinsic capacity to avoid overfitting, in line with the theoreticas results in Section 2.

In Section 2 we have proved that, on the regression interval, the derivative of the rectified MLP, the typical DC-MLP, has a moderate variation, in contrast to the derivative of the logistic MLP, the typical SQ-MLP, which is potentially unbounded. We have also proved that overfitting regression functions have a potentially unbounded derivative. These theoretical results would explain, at least partially, the empirical results in Section 3.

´All in all, DC-MLPs could result very useful for practical purposes based on one-dimensional nonlinear regression: they show no overfitting, avoiding the use of any additional regularization technique. We expect that DC-MLPs will ease the use of nonlinear regression for practical purposes. As a matter of further research, we plan to extend our analysis to the multidimensional case.

Acknowledgements

We thank the editor and reviewers for their constructive and stimulating suggestions. This research work has been supported by the following projects:

- TED2021-129162B-C22, funded by the Recovery and Resilience Facility (Plan de Recuperación, Transformación y Resiliencia - PRTR) program from the NextGenerationEU Plan of the European Union and the the Spanish Research Agency (Agencia Estatal de Investigación).
- PID2021-128362OB-I00, funded by the Spanish Plan for Scientific and Technical Research and Innovation (Plan Estatal de Investigación Científica y Técnica y de Innovación 2021-2023) of the Spanish Research Agency (Agencia Estatal de Investigación).

References

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al., Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, IEEE Signal processing magazine 29 (2012) 82–97.

[2] A. Graves, A. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2013, pp. 6645–6649.

[3] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder–decoder for statistical machine translation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1724–1734.

[4] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, in: J. Burstein, C. Doran, T. Solorio (Eds.), Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), Association for Computational Linguistics, 2019, pp. 4171–4186.

[5] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet classification with deep convolutional neural networks, Commun. ACM 60 (2017) 84–90.

[6] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, volume 27, Curran Associates, Inc., 2014, pp. 2672–2680.

[8] H.-Z. Li, S. Guo, C.-J. Li, J.-Q. Sun, A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm, Knowledge-Based Systems 37 (2013) 378–387.

[9] J. Song, C. E. Romero, Z. Yao, B. He, A globally enhanced general regression neural network for on-line multiple emissions prediction of utility boiler, Knowledge-based systems 118 (2017) 4–14.

[10] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of control, signals and systems 2 (1989) 303–314.

[11] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural networks 2 (1989) 359–366.

[12] R. Bellman, Dynamic programming, Science 153 (1966) 34–37.

[13] M. Belkin, D. Hsu, S. Ma, S. Mandal, Reconciling modern machine-learning practice and the classical bias–variance trade-off, Proceedings of the national academy of sciences 116 (2019) 15849–15854.

[14] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.

[15] Q. Nguyen, M. Hein, The loss surface of deep and wide neural networks, in: D. Precup, Y. W. Teh (Eds.), Proceedings of the 34th International Conference on Machine Learning, volume 70 of *Proceedings of Machine Learning Research*, PMLR, 2017, pp. 2603–2612.

[16] Y. Bengio, N. Roux, P. Vincent, O. Delalleau, P. Marcotte, Convex neural networks, in: Y. Weiss, B. Schölkopf, J. Platt (Eds.), Advances in Neural Information Processing Systems, volume 18, MIT Press, 2005, pp. 123–130.

[17] B. Amos, L. Xu, J. Z. Kolter, Input convex neural networks, in: D. Precup, Y. W. Teh (Eds.), Proceedings of the 34th International Conference on Machine Learning, volume 70 of *Proceedings of Machine Learning Research*, PMLR, 2017, pp. 146–155.

[18] S. Sivaprasad, A. Singh, N. Manwani, V. Gandhi, The curious case of convex neural networks, in: N. Oliver, F. Pérez-Cruz, S. Kramer, J. Read, J. A. Lozano (Eds.), Machine Learning and Knowledge Discovery in Databases. Research Track, Springer International Publishing, Cham, 2021, pp. 738–754.

[19] P. Sankaranarayanan, R. Rengaswamy, CDiNN – convex difference neural networks, Neurocomputing 495 (2022) 153–168.

[20] H. Tuy, Convex Analysis and Global Optimization, Springer Optimization and Its Applications, 2 ed., Springer International Publishing, Basel, Switzerland, 2016.

[21] H. A. Le Thi, T. Pham Dinh, DC programming and DCA: thirty years of developments, Mathematical programming 169 (2018) 5–68.

[22] T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 785–794.

[23] A. S. Minhas, S. Singh, A new bearing fault diagnosis approach combining sensitive statistical features with improved multiscale permutation entropy method, Knowledge-based systems 218 (2021) 106883.

[24] Y. Wang, Y. Guo, Forecasting method of stock market volatility in time series data based on mixed model of ARIMA and XGBoost, China communications 17 (2020) 205–221.

[25] M. M. H. Shandhi, W. H. Bartlett, J. A. Heller, M. Etemadi, A. Young, T. Plötz, O. T. Inan, Estimation of instantaneous oxygen uptake during exercise and daily activities using a wearable cardio-electromechanical and environmental sensor, IEEE Journal of biomedical and health informatics 25 (2021) 634–646.

[26] D. Specht, A general regression neural network, IEEE Transactions on neural networks 2 (1991) 568–576.

[27] F. Stulp, O. Sigaud, Many regression algorithms, one unified model: A review, Neural networks 69 (2015) 60–79.

[28] S. O. Haykin, Neural Networks and Learning Machines, 3 ed., Pearson, Upper Saddle River, NJ, 2008.

[29] E. Simhayev, G. Katz, L. Rokach, Integrated prediction intervals and specific value predictions for regression problems using neural networks, Knowledge-based systems 247 (2022) 108685.

[30] P. Morala, J. A. Cifuentes, R. E. Lillo, I. Ucar, Towards a mathematical framework to inform neural network modelling via polynomial regression, Neural networks 142 (2021) 57–72.

[31] M. Belkin, Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation, Acta numerica 30 (2021) 203–248.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, Journal of machine learning research 15 (2014) 1929–1958.

[33] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of computational physics 378 (2019) 686–707.

[34] H. H. Nguyen, T. Zieger, R. D. Braatz, R. Findeisen, Robust control theory based stability certificates for neural network approximated nonlinear model predictive control, IFAC-PapersOnLine 54 (2021) 347–352. 7th IFAC Conference on Nonlinear Model Predictive Control NMPC 2021.

[35] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, D. K. Duvenaud, Neural ordinary differential equations, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 31, Curran Associates, Inc., 2018, pp. 6571–6583.

[36] I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, 2015. doi:10.48550/arXiv.1412.6572. arXiv:1412.6572.

[37] A. Kurakin, I. J. Goodfellow, S. Bengio, Adversarial examples in the physical world, in: Artificial Intelligence Safety and Security, Chapman and Hall/CRC, 2018, pp. 99–112.

[38] T. M. Apostol, Mathematical Analysis, Addison-Wesley series in mathematics, 2 ed., Pearson, Upper Saddle River, NJ, 1974.

[39] O. Calin, Deep Learning Architectures: A Mathematical Approach, Springer Nature, Cham, Switzerland, 2020.

[40] J.-B. Hiriart-Urruty, C. Lemarechal, Fundamentals of Convex Analysis, Grundlehren Text Editions, 1 ed., Springer, Berlin, Germany, 2001.

[41] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, et al., PyTorch: An imperative style, high-performance deep learning library, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems 32, volume 32, Curran Associates, Inc., 2019, pp. 8024–8035.

[42] D. Dua, C. Graff, UCI machine learning repository, 2017. URL: http://archive.ics.uci.edu/ml.

[43] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, R. Fergus, Regularization of neural networks using dropconnect, in: International conference on machine learning, PMLR, 2013, pp. 1058–1066.

[44] H. Fanaee-T, Bike Sharing Dataset, UCI Machine Learning Repository, 2013.

[45] O. Akbilgic, ISTANBUL STOCK EXCHANGE, UCI Machine Learning Repository, 2013.

[46] D. Stolfi, Parking Birmingham, UCI Machine Learning Repository, 2019.

[47] R. Quinlan, Auto MPG, UCI Machine Learning Repository, 1993.

[48] Productivity Prediction of Garment Employees, UCI Machine Learning Repository, 2020.

[49] S. Vito, Air Quality, UCI Machine Learning Repository, 2016.

[50] S. Chen, Beijing PM2.5 Data, UCI Machine Learning Repository, 2017.

[51] AI4I 2020 Predictive Maintenance Dataset, UCI Machine Learning Repository, 2020.

[52] S. Matzka, Explainable artificial intelligence for predictive maintenance applications, in: 2020 Third International Conference on Artificial Intelligence for Industries (AI4I), 2020, pp. 69–74.

[53] M. Aceves-Fernandez, EEG Steady-State Visual Evoked Potential Signals, UCI Machine Learning Repository, 2018.

[54] F. Zamora-Martínez, P. Romeu, P. Botella-Rocamora, J. Pardo, On-line learning of indoor temperature forecasting models towards energy efficiency, Energy and Buildings 83 (2014) 162–172.

[55] P. Romeu-Guallart, F. Zamora-Martinez, SML2010, UCI Machine Learning Repository, 2014.

A. Minimal implementation of a DC-MLP in Pytorch - example

```
class DC_MLP(nn.Module):
    def __init__(self, D_in, H, D_out):
        super(DC_MLP, self ).__init__()

        self.activ  = nn.Softplus()
        self.fc1    = nn.Linear(D_in, H)
        self.fc2    = nn.Linear(H, D_out)

    def forward(self, x):
        out = self.fc1(x)
        out = self.activ(out)
        out = self.fc2(out)
        return out
```

Note that, if in the previous class a squashing activation function is used (logistic, hyperbolic tangent, etc.), then, the DC-MLP becomes a SQ-MLP, by definition.