Abhirath Bhuvanesh, Joseph Ntaimo, Lidia Smith

Exploring machine learning techniques for solving differential equations

To gain insight into techniques from machine learning, like support vector machines (SVMs) or convolutional neural networks (CNN) - a type of deep neural network, we propose to apply the techniques in solving differential equations and to analyze their efficiency. There is some published work in this direction, but we will consider a variation of the SVMs and compare the performance with that of using deep neural networks to approximate solutions to some simple differential equation for which exact solutions are known.
After understanding the techniques, more complicated differential equations that do not have analytic solutions, like those modeling fluid flow, can be tacked with the new approximating algorithms.
Recent progress in the fields of artificial intelligence (AI) or machine learning has led to the creation of algorithms that can solve problems in object recognition or natural language processing that were intractable a few decades ago.
These algorithms, even though very successful are not well understood from a mathematical point of view. Simple blocks, like a linear function of the input, followed by a non-linear transformation that could be as simple a replacing the negative part by zero (called an ReLU - rectified linear unit) are used to create multiple layers and hidden units, meant to abstract key features from the input data presented to the network. The evaluation of features is followed by a prediction step in which a category is selected (in a classification problem) or a function is estimated.
To approximate solutions to differential equations, a function linear in parameters is used as model. This is typical when estimating functions based on SVMs.

$$f(x) = \sum_{i=1}^{n} w_i \varphi_i(x) + b = \mathbf{w}^T \boldsymbol{\varphi}(x) + b$$

where x is the input data, $\mathbf{w}$ is a vector of weights, and the function $\boldsymbol{\varphi}$ maps the inputs x into the feature space; the term, b, called bias, is added to the linear combination of features $\varphi_(x)$ with weights w_i .
We will work with $\boldsymbol{\varphi}$ that satisfies a kernel property $\boldsymbol{\varphi}^T(x_j)\boldsymbol{\varphi}(x_i) = K(x_i, x_j) = \exp\left(-\frac{|x_i - x_j|^2}{2\sigma^2}\right)$. Other kernels can be used, but the Gaussian kernel is a popular one.
The unknowns in the model are the weights w_i and the bias term b. The training phase of the algorithm is finding the best values for those parameters in order to predict the output values corresponding to new inputs that have not been part of the training set of inputs.