USE OF LINEAR ALGEBRA AND PARTIAL DERIVATIVES IN SUPERVISED LEARNING (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

[1]Prerana Misra, [2]Avik Mukherjee & [3]Anish Pyne

[1]CSE (2nd Year), Institute Of Engineering & Management, Kolkata
Email Id: Preranamisra24@Gmail.Com
[2]CSE (2nd Year), Institute Of Engineering & Management, Kolkata
Email Id: Mukherjeeavik52@Gmail.Com
[3]EEE (2nd Year), Institute Of Engineering & Management, Kolkata
Email Id: Pyneanish1@Gmail.Com

Abstract

When we talk about new technologies and the advancement in the field of Computer Science, the first thing that comes to our mind is Artificial Intelligence and Machine Learning. Artificial Intelligence has seen resurgence in the 21st century because of its ability to mimic functions done by human intelligence like "problem solving" and "learning". It is slowly becoming the area of interest of the new generation because of its modern capabilities which even human intelligence struggle to perform like competing at highest level in strategic game systems, intelligent routing, operating cars autonomously and simulations. Artificial Intelligence may look easy but there are several tools involved in making it successful. One of the main tool is "Statistical Methods". Linear algebra and Partial Differential Equations have become the base of this field. The objective of our paper is to throw light on how Statistical Methods and Mathematical optimization provide the base for the working of Supervised Learning. Over years, algorithms inspired by Partial Differential Equations (PDE) and Linear Algebra have had an immense impact on many processing and autonomously performed tasks that involve speech, image and video data. Image processing tasks and intelligent routing done using PDE models has lead to ground-breaking contributions. The reinterpretation of many modern machine capabilities like artificial neural networks through PDE lens has been creating multiple celebrated approaches that benefit a vast area. In this paper, we have established some working of these methods in different sub-fields of Artificial Intelligence. Guided by well-established theories we demonstrate new insights and algorithms for Supervised Learning and demonstrate the competitiveness of different numerical experiments used in the sub-fields. Not only will we see the wide application of Artificial intelligence but also its ability to slowly replace human works leading to unemployment which are part of its limitation. This research will provide wider insights into the multiple mathematical processes which acts as roots to make the field of Computer Science interesting and successful.

Keywords: *Computer Science, Artificial Intelligence, Partial Differentiation, Linear Algebra.*

INTRODUCTION

Other forms of differentiation like Automatic differentiation (AD) also called algorithmic differentiation are used as derivatives to evaluate numeric functions expressed as computer

programs. Automatic differentiation has its applications in areas including atmospheric sciences, engineering design optimization and computational fluid dynamics. Artificial intelligence is divided into sub-fields. These sub-fields fail to communicate with each other but do not lose their importance individually. These are based on technical considerations such as use of particular tools (e.g. "logic" or "artificial neural networks"), striving towards particular goals (e.g. "machine learning" or "robotics"), or philosophical differences. AI research aims to excel in reasoning, planning, learning, knowledge representation, natural language processing, perception and the ability to move and manipulate objects. A part in Tesla's theorem says that "AI is whatever hasn't been done yet". Therefore, when machines become increasingly capable, tasks considered to require "intelligence" are removed from AI known as AI effect and more difficult tasks are taken up. AI was founded as an academic discipline in 1956 but soon lost its popularity due to lack of funding (known as "AI winter"). Again, in the 21st century it gained back its popularity due to new approaches, success and renewed funding. AI needs to implement the representation of a couple of things such as categories, objects, relations, properties and so on. All of them are connected to mathematics, as well as act as very adequate illustrative examples. AI problems can be classified into two types, Search problems and Representation problems and their interconnected models and tools are Logics, Rules, Frames and Nets. AI creates an admissible model for the human knowledge. In this paper we review Linear algebra and Partial Derivative from a Supervised learning perspective, covering its origins, applications in Artificial Intelligence, and methods of implementation. The enthusiastic practitioner who is interested to learn more about the magic behind successful machine learning and AI algorithms currently faces a daunting set of pre-requisite knowledge:

- Programming languages and data analysis tools.
- Large-scale computation and the associated frameworks.
- Mathematics and statistics and how machine learning builds on it.

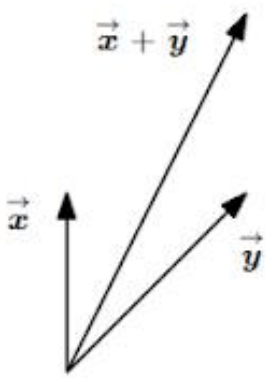
This research paper brings the mathematical foundations of basic supervised learning concepts to the fore and collects the information in a single place so that the skills gap is narrowed. In section 1 we discuss about the mathematical foundations of the four pillars of Machine Learning and Artificial Intelligence.

1.LINEAR ALGEBRA

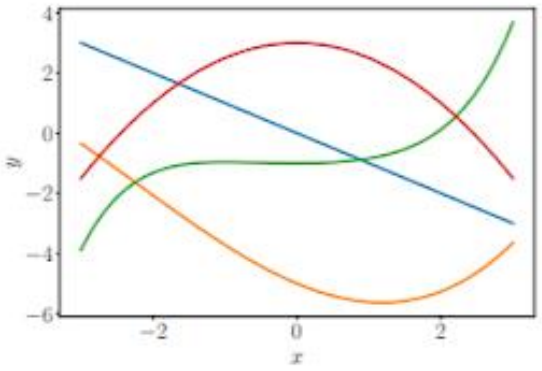
Linear Algebra is commonly known as the study of vectors. It also contains rules to manipulate vectors. The different forms of vectors are:

- Geometric Vectors- Geometric vectors are vectors which can be drawn at least in two dimensions. Two geometric vectors $\rightarrow x, \rightarrow y$ can be added, such that $\rightarrow x + \rightarrow y = \rightarrow z$ is another geometric vector.
- Polynomial Vectors- Polynomial vectors are abstract concepts in which two polynomials are added together, which results in another polynomial which can be multiplied by a scalar $\lambda \in R$, and the result is a polynomial as well.

- Elements of Rn (tuples of n real numbers) are vectors. Rn is more abstract than polynomials. Therefore, it can also be considerd as vectors.



(a) Geometric vectors.



(b) Polynomials.

Linear algebra focuses on the similarities between these vectors.

Linear algebra plays an important role in machine learning and artificial intelligence. Linear algebra is important because it will be later combined with other basic aspects.

b) Matrices: -

Matrices play a central role in linear algebra. They can be used to compactly represent systems of linear equations, but they also represent linear functions (linear mappings) . With m, n ∈ N a real-valued (m, n) matrix A is an m·n-tuple of elements aij , i = 1, . . . , m, j = 1, . . . , n, which is ordered according to a rectangular scheme consisting of m rows and n columns:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad a_{ij}$$

By convention (1, n)-matrices are called rows and (m, 1)-matrices are called column columns. These special matrices are also called row/column vectors. By stacking its columns, a matrix A can be represented as a long vector a. re-shape A ∈ R 4×2 a ∈ R 8 R^(m×n) is the set of all real-valued (m, n)-matrices. A ∈ R^(m×n) can be equivalently represented as a ∈ R^(mn) by stacking all n columns of the matrix into a long vector.

1.1. Use of linear algebra to create dataset and data files for AI: -

Dataset and data files are table-like set of numbers where each column represents a feature of the observation and where each row represents an observation.

Example the Iris flowers dataset

The data is a matrix which is a part of linear algebra.

When we split the data into inputs and outputs to fit a supervised learning model, such as the measurements and the flower species, we have a matrix(X) and a vector(y). The data is vectorized as each row has the same length i.e. the same number of columns.

1.2. In images and photographs: -

A photo is an example of matrix from linear algebra. Each image is itself a table structure with a width and height and one pixel in each cell for black and white images or three pixels for color images. Image operations such as cropping, scaling, shearing and so on are all described using examples of linear algebra.

1.3. In data encoding: -

Sometimes in supervised learning, we need to work with categorical data. Categorical variables are encoded to make them easier to work with. One of the process is called one hot encoding. One hot encoding is where a table is created to represent the variable with one column for each category and a row for each example in dataset. A check, or each one-value, is added in the column for the categorical value for a given row, and a zero-value is added to all other columns.

1.4. In Principal Component Analysis: -

A dataset may have thousands of columns. Modelling data with many columns is quite difficult, and models built from irrelevant features are less skilful. Methods for automatically reducing the number of columns of a dataset are called dimensionality reduction, also the most popular method is called principal component analysis. This method is used in AI to create projections of high-dimensional data for visualization. Matrix factorization acts as the base of principal component analysis.

1.5. In Singular Value Decomposition: -

As stated above dimensionality can be automatically reduced in AI using Linear Algebra. Another method of this is singular-value decomposition. It is also a matrix factorization method. It has its application in selection, noise reduction, visualization and more.

1.6. In regularization: -

We often tend to seek the simplest possible methods that achieve the best skill. Simple models are used to generalize specific examples to unseen data. A technique that is often used to encourage a model to minimize the size of coefficients while it is being fit on data is called regularization. These forms of regularization are a measure of the magnitude or length of the coefficients as a vector and are method lifted directly from linear algebra called the vactor norm.

2. Partial Derivatives: -

2.1. Differential Equations in Residual Networks: -

The abstract goal of machine learning and AI is to find a function f : R n × R p → R m such that f(·, θ) accurately predicts the result of an observed phenomenon (e.g., the class of an image, airplane bird car cat deer dog horse monkey ship truck true label Hamiltonian CNN Parabolic CNN second-order CNN airplane bird car cat deer dog horse monkey ship truck true label Hamiltonian CNN Parabolic CNN second-order CNN true label Hamiltonian CNN Parabolic CNN second-order CNN true label Hamiltonian CNN Parabolic CNN second-order CNN true label Hamiltonian CNN Parabolic CNN second-order CNN true label Hamiltonian CNN Parabolic CNN second-order CNN airplane bird car cat deer dog horse monkey ship truck true label Hamiltonian CNN Parabolic CNN second-order CNN airplane bird car cat deer dog horse monkey ship truck true label Hamiltonian CNN Parabolic CNN second-order CNN airplane bird car cat deer dog horse monkey ship truck true label Hamiltonian CNN Parabolic CNN second-order CNN airplane bird car cat deer dog horse monkey ship truck true label Hamiltonian CNN Parabolic CNN second-order CNN Figure 1:

$$\mathbf{F}(\boldsymbol{\theta}, \mathbf{Y}) = \mathbf{K}_2(\boldsymbol{\theta}^{(3)})\sigma\left(\mathcal{N}(\mathbf{K}_1(\boldsymbol{\theta}^{(1)})\mathbf{Y}, \boldsymbol{\theta}^{(2)})\right). \qquad (1)$$

Classification results of the three proposed CNN architecture for four randomly selected test images from the STL10 dataset . The predicted and actual class probabilities are visualized using bar plots on the right of each image. While all networks reach a competitive prediction accuracy between around 74% and 78% across the whole dataset, predictions for individual images vary in some cases. a spoken word, etc.). The function is parameterized by the weight vector θ ∈ R p that is trained using examples. In supervised learning, a set of input features y1 , . . . , ys ∈ R n and output labels c1, . . . , cs ∈ R m is available and used to train the model f(·, θ). The output labels are vectors whose components correspond to the estimated probability of a particular example belonging to a given class. As an example, consider the image classification results in Fig. 1 where the predicted and actual labels are visualized using bar

plots. For brevity, we denote the training data by Y = [y1, y2, . . . , ys] ∈ R n×s and C = [c1, c2, . . . , cs] ∈ R m×s . In deep learning, the function f consists of a concatenation of nonlinear functions called hidden layers. Each layer is composed of affine linear transformations and pointwise nonlinearities and aims at filtering the input features in a way that enables learning. As a fairly general formulation, we consider an extended version of the layer used in [22], which filters the features Y as follows F(θ, Y) = K2(θ (3))σ N (K1(θ (1))Y, θ (2))

Here, the parameter vector, θ, is partitioned into three parts where θ (1) and θ (3) parameterize the linear operators K1(·) ∈ R k×n and K2(·) ∈ R kout×k , respectively, and θ (2) are the parameters of the normalization layer N . The activation function σ : R → R is applied component-wise. Common examples are σ(x) = tanh(x) or the rectified linear unit (ReLU) defined as σ(x) = max(0, x). A deep neural network can be written by concatenating many of the layers.

Theorem 1 If the activation function σ is monotonically nondecreasing, then the forward propagation through a parabolic CNN satisfies (6).

Proof 1 For ease of notation, we assume that no normalization layer is used, i.e., N(Y) = Y in (8). We then show that Fsym(θ(t),Y) is a monotone operator. Note that for all t ∈ [0,T] −(σ(K(t)Y)−σ(K(t)Y),K(t)(Y−Y)) ≤ 0. Where (·,·) is the standard inner product and the inequality follows from the monotonicity of the activation function, which shows that ∂tkY(t)−Y(t)k2 F ≤ 0. Integrating this inequality over [0,T] yields stability as in (6). The proof extends straightforwardly to cases when a normalization layer with scaling and bias is included.

One way to discretize the parabolic forward propagation (8) is using the forward Euler method. Denoting the time step size by δt > 0 this reads Yj+1 = Yj + δtFsym(θ(tj),Yj), j = 0,1,...,N −1, where tj = jδt. The discrete forward propagation of a given example y0 is stable if δt satisfies

max i=1,2,...,n|1 + δtλi(J(tj))|≤ 1, j = 0,1,...,N −1,

and accurate if δt is chosen small enough to capture the dynamics of the system. Here, λi(J(tj)) denotes the ith eigenvalue of the Jacobian of Fsym with respect to the features at a time point tj. If we assume, for simplicity, that no normalization layer is used, the Jacobian is J(tj) = −K>(θ(1)(tj)) D(tj)K(θ(1)(tj)), with D(t) = diagσ0K(θ(1)(t))y(t). If the activation function is monotonically nondecreasing, then σ0(·) ≥ 0 everywhere. In this case, all eigenvalues of J(tj) are real and bounded above by zero since J(tj) is also symmetric. Thus, there is an appropriate δt that renders the discrete forward propagation stable.

mislead deep networks by being barely noticeable to a human observer (e.g., [18, 37, 35]). To ensure the stability of the network for all possible weights, we propose to restrict the space of CNNs. As examples of this general idea, we present three new types of residual CNNs that are motivated by parabolic and first- and second-order hyperbolic PDEs, respectively. The construction of our networks guarantees that the networks are stable forward and, for the hyperbolic network, stable backward in time. Though it is common practice to model K1 and

K2 in (1) independently, we note that it is, in general, hard to show the stability of the resulting network. This is because, the Jacobian of F(θ,Y) with respect to the features has the form JYF = K2(θ) diag(σ0(K1(θY))) K1(θ),

where σ0 denotes the derivatives of the pointwise nonlinearity and for simplicity we assume N(Y) = Y. Even in this simplified setting, the spectral properties of JY, which impact the stability, are unknown for arbitrary choices of K1 and K2. As one way to obtain a stable network, we introduce a symmetric version of the layer in (1) by choosing K2 = −K> 1 in (1). To simplify our notation, we drop the subscript of the operator and define the symmetric layer Fsym(θ,Y) = −K(θ)>σ (N(K(θ)Y,θ)). It is straightforward to verify that this choice leads to a negative semi-definite Jacobian for any non-decreasing activation function. As we see next, this choice also allows us to link the discrete network to diff erent types of PDEs.

Numerical Experiments

We demonstrate the potential of the proposed architectures using the common image classification benchmarks STL-10 [13], CIFAR-10, and CIFAR-100 [28]. Our central goal is to show that, despite their modeling restrictions, our new network types achieve competitive results. We use our basic architecture for all experiments, do not excessively tune hyperparameters individually for each case, and employ a simple data augmentation technique consisting of random flipping and cropping.

Network Architecture. Our architecture is similar to the ones in [22, 9] and contains an opening layer, followed by several blocks each containing a few time steps of a ResNet and a connector that increases the width of the CNN and coarsens the images. Our focus is on the diff erent options for defining the ResNet block using parabolic and hyperbolic networks. To this end, we choose the same basic components for the opening and connecting layers. The opening layer increases the number of channels from three (for RGB image data) to the number of channels of the first ResNet using convolution operators with 3×3 stencils, a batch normalization layer and a ReLU activation function. We build the connecting layers using 1×1 convolution operators that increase the number of channels, a batch normalization layer, a ReLU activation, and an average pooling operator that coarsens the images by a factor

3.Drawbacks of Machine learning:

Limitation 1 — Ethics

Machine learning, a subset of artificial intelligence, has revolutionized the world as we know it in the past decade. The information explosion has resulted in the collection of massive amounts of data, especially by large companies such as Facebook and Google. This amount of data, coupled with the rapid development of processor power and computer parallelization, has now made it possible to obtain and study huge amounts of data with relative ease.

It is easy to understand why machine learning has had such a profound impact on the world, what is less clear is exactly what its capabilities are, and perhaps more importantly, what its limitations are. Yuval Noah Harari famously coined the term ‘dataism’, which refers to a

putative new stage of civilization we are entering in which we trust algorithms and data more than our own judgment and logic.

Whilst you may find this idea laughable, remember the last time you went on vacation and followed the instructions of a GPS rather than your own judgment on a map — do you question the judgment of the GPS? People have literally driven into lakes because they blindly followed the instructions from their GPS.

The idea of trusting data and algorithms more than our own judgment has its pros and cons. Obviously, we benefit from these algorithms, otherwise, we wouldn't be using them in the first place. These algorithms allow us to automate processes by making informed judgments using available data. Sometimes, however, this means replacing someone's job with an algorithm, which comes with ethical ramifications. Additionally, who do we blame if something goes wrong?

The most commonly discussed case currently is self-driving cars — how do we choose how the vehicle should react in the event of a fatal collision? In the future will we have to select which ethical framework we want our self-driving car to follow when we are purchasing the vehicle?

If my self-driving car kills someone on the road, whose fault is it?

Whilst these are all fascinating questions, they are not the main purpose of this article, Clearly, however, machine learning cannot tell us anything about what normative values we should accept, i.e. how we should act in the world in a given situation. As David Hume famously said, one cannot 'derive an ought from an is'.

CONCLUSION:

Research in this field helped in rapid prototyping and development cycle for testing new models and ideas using mathematical knowledge. We expect this to be the core of Machine Learning and Artificial Intelligence for the foreseeable future. It is an exciting time for working with AI and mathematical topics and there are many opportunities for bringing advanced techniques and expertise in this field. By this new approach we defend, Computer Science occupies, partially and in a natural way, the role Physics and its problems have played as support of mathematical reasoning, a fact in the past two centuries (although Physics do not disappear from the view, being a necessary aid). We propose showing such Methods through the parallel study of Mathematics and Computer Science foundations. Other Computer Science subfields could be carriers of this method too, but perhaps AI is the current better choice, given its characteristics, which practically coincide with many mathematical techniques and objectives. The creative learning permits to understand the development and practice of creativity. The possibility of founding new solutions is one specific characteristic of the creative process. It may consists in the art of formulate questions to obtain ideas, increasing capacities, defying the current conventionalism in the educative world. So, the benefits of such an innovative educative method must consist in a more progressive regard of Mathematical

Education in modern times, with the final purpose of producing adaptive and creative minds, capable of solving new problems and challenges.

Discussion and Outlook:

In this paper, we establish a link between deep residual convolutional neural networks and PDEs. The relation provides a general framework for designing, analysing, and training those CNNs. It also exposes the dependence of learned weights on the image resolution used in training. Exemplarily, we derive three PDE-based network architectures that are forward stable (the parabolic network) and forward-backward stable (the hyperbolic networks). It is well-known that diff erent types of PDEs have diff erent properties. For example, linear parabolic PDEs have decay properties while linear hyperbolic PDEs conserve energy. Hence, it is common to choose diff erent numerical techniques for solving and optimizing diff erent kinds of PDEs. The type of the underlying PDE is not known a-priori for a standard convolutional ResNet as it depends on the trained weights. This renders ensuring the stability of the trained network and the choice of adequate time-integration methods difficult. These considerations motivate us to restrict the convolutional ResNet architecture a-priori to discretization of nonlinear PDEs that are stable. In our numerical examples, our new architectures lead to an adequate performance despite the constraints on the networks. In fact, using only networks of relatively modest size, we obtain results that are close to those of state-of-the-art networks with a considerably larger number of weights. This may not hold in general, and future research will show which types of architectures are best suited for a learning task at hand. Our intuition is that, e.g., hyperbolic networks may be preferable over parabolic ones for image extrapolation tasks to ensure the preservation of edge information in the images. In contrast to that, we anticipate parabolic networks to perform superior for tasks that require filtering, e.g., image denoising. We note that our view of CNNs mirrors the developments in PDE-based image processing in the 1990s. PDE-based methods have since significantly enhanced our mathematical understanding of image processing tasks and opened the door to many popular algorithms and techniques. We hope that continuous models of CNNs will result in similar breakthroughs and, e.g., help streamline the design of network architectures and improve training outcomes with less trial and error.

REFERENCES:

[1] L. Ambrosio and V. M. Tortorelli. Approximation of Functionals Depending on Jumps by Elliptic Functionals via Gamma-Convergence. Commun. Pure Appl. Math., 43(8):999–1036, 1990.

[2] U. Ascher. Numerical methods for Evolutionary Diff erential Equations. SIAM, Philadelphia, USA, 2010.

[3] U. Ascher, R. Mattheij, and R. Russell. Numerical Solution of Boundary Value Problems for Ordinary Diff erential Equations. SIAM, Philadelphia, Philadelphia, 1995.

[4] Y. Bengio et al. Learning deep architectures for AI. Found. Trends Mach. Learn., 2(1):1–127, 2009.

[5] Y. Bengio, P. Simard, and P. Frasconi. Learning Long-Term Dependencies with Gradient Descent Is Difficult. IEEE Transactions on Neural Networks, 5(2):157–166, 1994.

[6] L. T. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders, editors. Real-time PDE-constrained Optimization. Society for Industrial and Applied Mathematics (SIAM), 2007.

[7] A. Borz`ı and V. Schulz. Computational optimization of systems governed by partial diff erential equations, volume 8. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2012.

[8] T. F. Chan and L. A. Vese. Active contours without edges. IEEE Trans. Image Process., 10(2):266–277, 2001.

[9] B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham. Reversible architectures for arbitrarily deep residual neural networks. In AAAI Conference on AI, 2018.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. arXiv.org, Dec. 2014.

[11] E. Haber and L. Ruthotto. Stable architectures for deep neural networks. Inverse Probl., 34:014004, 2017.