Partial Differential equations via Data Discovery & Sparse Optimization

Research paper by Hayden Schaeffer-2016

Published at THE ROYAL SOCIETY

Submitted By:

Pranav S

Aditya Kr Gautam

Noorshaba

Priyanka Soni

Introduction

- Objective: To learn underlying PDE only using data. For this we need to :

  - Identify terms of PDE (Form of model)

  - Approximate coefficient of terms (Parameter estimation)

- Approach: 1. Symbolic Regression 2. Sparse regression
- Rather than using symbolic regression, a sequential thresholded least-squares algorithm is applied.
- In this work, **a sparse optimization method** is proposed for identifying the underlying PDE to a given data set. Learning algorithm for this uses sparse optimization for feature selection and parameter estimation.
- The regression approach balances between the efficiency of the candidate model and the accuracy of the fit.

Sparse Optimization

Sparse optimization theory deals with sparse solutions for systems of linear equations. Techniques for finding these solutions and exploiting them in applications have found wide use in image processing, signal processing, machine learning, medical imaging, and more. Sparsification can also be seen as some form of regularization, and may improve model quality by effectively reducing noise in the model, Modern networks are computationally expensive to use. Furthermore, training such deep neural models becomes increasingly expensive. Thus, it is important to investigate sparsity during the training process to manage the costs of training.

The L1 norm is often used as a proxy for sparsity, because it allows for the construction and implementation of feasible numerical schemes.

The linear system is typically solved via an L1 regularized least-squares minimization and (in some cases) can be shown to exactly recover the sparse solution.

L1 Norm and L1 Regularisation

- L1 Norm: $||A||_1 = \Sigma|A_i|$
- LASSO Regression is commonly used in data fitting, with the L1 norm on the unknown coefficients.
- It is basically L1 Regularization:

$$1/n\ \Sigma(y_i - h_\theta(x_i))^2 + \lambda\ \Sigma|\theta_i|$$

Here λ is balancing parameter.

- The *L*1 norm is used to penalize the number of non-zero coefficients in order to promote coefficient sparsity.

Motivation and Model

In the research paper they considered the case of viscous Burgers, equation and created a general approach to solve PDEs.

With the viscous Burger equation given by:

$$u_t = (u^2/2)_x + \nu u_{xx}$$

Here they considered an unknown form of equation as the solution and approached with a general form of 2nd order PDE with quadratic nonlinearity. The r.h.s. was obtained by approximating a general second-order evolution equation $u_t = F(u, u_x, u_{xx})$ via a second-order Taylor expansion.

$$u_t = \alpha_1 + \alpha_2 u + \alpha_3 u^2 + \alpha_4 u_x + \alpha_5 u_x^2 + \alpha_6 u u_x + \alpha_7 u_{xx} + \alpha_8 u_{xx}^2 + \alpha_9 u u_{xx} + \alpha_{10} u_x u_{xx}.$$

$$\Downarrow$$

$$u_t = [1 \ u \ u^2 \ u_x \ u_x^2 \ uu_x \ u_{xx} \ u_{xx}^2 \ uu_{xx} \ u_x u_{xx}] \cdot \alpha.$$

Considering $f_l(t)$ as the feature vector, the collection of feature vectors defines the feature matrix, i.e. $F_u(t) = [f_l(t).]$

$V(t)$ is the velocity vector. Here we get this problem liner in α:

$$V(t) = F_u(t)\,\alpha.$$

This equation is solved using L^1 regularized least square minimisation given by:

$$\min_\alpha \tfrac{1}{2} \Sigma \| V(t_k) - F_u(t_k)\,\alpha \|_2^2 + \lambda \| \alpha \|_1.$$

Here $\lambda > 0$ is a balancing parameter. The L1 regularization was choose to promote sparsity in the vector α, with the modelling assumption that the underlying dynamics were governed by a few terms.

Numerical Method and Algorithm

- From the inference obtained by solving viscous Burgers' equation we move on to a generalised method.
- If we have the data w(x,t) at (possibly non-uniform) grid points xj for $1 \leq j \leq N$ and time stamps tk for $1 \leq k \leq M$, we approximate the underlying evolution equation $w_t = G(w)$.
- Then we will make a distinction between the given (discrete) data w(x, t) and the computed variable u(x, t) that is generated by a particular PDE.
- Here initially we compute a numerical approximation to the spatial derivatives of w(x, t), then construct a collection of feature vectors.
- Finally the coefficient of the terms are computed using least squares method.

$$\min_{\alpha} \tfrac{1}{2} \Sigma \| V(t_k) - F_u(t_k)\, \alpha \|_2^2 + \lambda \| \alpha \|_1.$$

Numerical Method and Algorithm

The minimizer satisfies the following inclusion relation

$$\sum_{k=1}^{M} (F_w^T(t_k)F_w(t_k)).\alpha + \sum_{k=1}^{M} F_w^T(t_k)V(t_k) + \lambda\partial\|\alpha\|_1$$

This equation is solved using the Douglas-Rachford algorithm. We define two terms:

$$H_1(\alpha) = \lambda\|\alpha\|_1, \qquad H_2(\alpha) = ½ \sum_{k=1}^{M} \|V(t_k) - F_w(t_k)\,\alpha\|^2_2$$

The proximal operators for these functions can be written as

$$\mathrm{prox}_{\gamma\,H1}(x) = \mathrm{argmin}_y\ \gamma\,\lambda\|y\|_1 + ½\,\|x - y\|^2 = \max(|x| - \gamma\,\lambda,\ 0)\ \mathrm{sign}(x)$$

The Douglas–Rachford iteration is defined by the following two-step method

$$\tilde{\alpha}^{k+1} = (1 - \mu/2)\,\tilde{\alpha}^{k} + \mu/2\ \ \mathrm{rprox}_{\gamma\,H2}(\mathrm{rprox}_{\gamma\,H1}(\tilde{\alpha}^{k}))$$

$$\alpha^{k+1} = \mathrm{prox}_{\gamma\,H1}(\tilde{\alpha}^{k+1})$$

If the restriction of the matrix $\sum_{k=1}^{M} F_w^T(t_k)F_w(t_k)$ onto the support set I is well conditioned, we can refine the solution α^* by a second optimization over all vectors with the same support.

$$\min_{\mathrm{supp}(\alpha^*) = I} ½ \sum_{k=1}^{M} \|V(t,) - F,(t,)\,\alpha\|_2^2$$

This step removes any bias in the value of the coefficients while retaining the features.

Method of Approach

Step 01

Initially $w(x, t)$ will be given, from that we have to calculate all spatial derivatives (up to some order) using the **spectral method.**

Step 02

Construct feature vectors via algebraic equations of the spatial derivatives. Normalize feature vectors to have maximum value 1.

Step 03

Use the Douglas–Rachford algorithm, equation, to solve the L1 least-squares Problem and extract the support of the vector.

Step 04

(Optional)
If the matrix M $\Sigma^{M}_{k=1} F^{T}_{w}(t_k)\, F_{w}(t_k)$ restricted to the support set I is well conditioned, then refine the solution of least square equation by solving equation

Simulation and Numerical Experiments:

- To test the method,
  - Data are simulated using the Crank–Nicolson method. The simulated data $u(x,t)$ is assigned to the variable $w^0(x,t)$.
  - The noise-free velocity w^0_t is computed by a first-order backward difference:

    $w^0_t(x_j, t_k) \approx [w^0(x_j, t_k) - w^0(x_j, t_k-1)]/ dt$, where $dt > 0$ is the time step.

  - The computed velocity w^0_t is corrupted by additive Gaussian noise.

    The noisy velocity is assigned to w_t.
  - The noise level is displayed for all examples by measuring the ratio between the L2 norm of the noise and the L2 norm of the data, i.e.

    $\text{noise level} = (\|w_t - w^0_t\|_2 \times 100\%)/ \|w^0_t\|_2$

- All spatial derivatives are approximated using **the spectral method,** and features are computed via direct products. All features are computed using $w^0_t(x_j,t_k)$.
- In all examples, the Douglas–Rachford algorithm parameters are set to $\gamma = 1/2$, $\mu = 1$, and the maximum number of iterations to 5000. Convergence of the iterates is achieved before the maximum iteration bound is reached.

Examples and Applications:

1. Viscous Burgers' equation:

$$u_t = (u^2/2)_x + \nu u_{xx}$$

where $\nu > 0$ is viscosity and $x \in [0,1]$

- Data are simulated using the Crank–Nicolson method.
- The balance parameter is fixed to $\lambda = 500$.
- The feature space is made up of the terms from a third-order Taylor expansion using up to second-order derivatives.
- The proposed algorithm is applied to the data with various levels of noise. For all noise levels, 2.0%, 19.5%, and 94.5%, the coefficients are identified to be within 1% in relative error.

This example shows that the method seems robust to noise.

2. Inviscid Burgers' equation

- To generate the solution to the inviscid Burgers' equation, we simulate the viscous equation with $0 < \nu << 1$ and $x \in [0, 1]$.
- Data are simulated using the Crank–Nicolson method.
- The balance parameter is fixed to $\lambda = 500$.
- The feature space is made up of the terms from a third-order Taylor expansion using up to second-order derivatives.
- The algorithm is applied to the data, with various levels of noise. For all noise levels, 1.5%, 15.0% and 73.7%, the coefficients are identified to within 2% in relative error.
  As the noise level increases, the identification of the correct features stays stable. This shows the robustness of the method to noisy data.
- In both examples, the dynamics are learned from $t \in [0, 0.05]$ but the comparisons are done at $T = 0.1$, showing the possibility to **extrapolate** dynamics and **predict** critical events (such as shock formation).

3. Swift–Hohenberg equation

- The Swift–Hohenberg equation is a scalar evolution equation with fourth-order derivatives:

$$u_t = -(1+\Delta)^2 u + \alpha u + \Box u^2 - u^3$$

- It is a model for thermal convection and solutions form various patterns depending on the choice of α and β.
  E.g. $(\alpha, \beta) = (0.1, 0)$, yield curves, $(\alpha, \beta) = (0.5, 0)$, yields a coral-like structure, $(\alpha, \beta) = (0.05, 0.5)$ yields a hexagonal structure.

- The terms are all identified correctly and the coefficients are within 2.8% of the true value.

- This example shows that, although the feature space contains redundancies (because $\Delta u = u_{xx} + u_{yy}$), the method is able to find the underlying dynamics.

- It demonstrates the proposed method's ability to identify the correct parameter regimes for the various patterns.

4. Cahn–Hilliard equation

$$u_t = -\gamma\Delta^2 u + \Delta(u^3 - u)$$

In this example, we will assume that the model takes the form: $u_t = \Delta F(u)$.

The data are simulated up to T = 2 using the Crank–Nicolson method with $\gamma = 0.5$.

Similar to example 3 the learning is stable with respect to redundancies in the feature space. The input data to the learning method use uniform time steps with a total of 385, 85 and 25 time stamps.

In all cases, the noise level is kept at 5.0%.

The Learned Coefficient show that the method is robust the number of time stamps used in the learning process.

This suggests that the proposed method is robust to the **size of the data**.

5. Navier–Stokes, low Reynolds number

The vorticity formulation for the two-dimensional Navier–Stokes equation is:

$$u_t+(\nabla^{\perp}\Delta^{-1}u).\ \nabla u = \Delta u/Re$$

The vorticity is simulated using the Crank–Nicolson method with various Reynolds numbers (Re > 0).
The terms are all identified correctly and the coefficients are within 1% of the true value
Similar to example 3&4, the learning is stable with respect to redundancies in the feature space.

The Reynolds number is chosen to be small enough so that the dynamics are dominated by viscous flow.

The coefficient shows that the method identifies the viscous flow model, which is the true dominant behaviour of the data.

6. Navier–Stokes, High Reynolds number

Using the vorticity formulation

$u_t+(\nabla^{\perp}\Delta^{-1}u).\ \nabla u = 0,$

we examine the turbulent flow case (i.e. Re>>1].

The data is stimulated with very high Reynold number .

It is known that the limiting flow (as the Reynolds number goes to infinity) of the Navier–Stokes flow on the torus is Euler's equation.

The method identifies the correct terms (with respect to Euler's equation) and the coefficients are within 1% of the true value.

The method picks out the coefficients to the intrinsic dynamics and not the equations used to simulate the data.

7. Homogenization of convection equation

Consider the convection equation with oscillatory velocity:

$u^{\epsilon}_{t} + a(x/\epsilon).\nabla u^{\epsilon}=0,$

with initial data u0(x, x/e) and x ∈ [0, 2π].

The data are simulated using 1 048 576 grid points with e=½ & e=1/64.

Step 4 of the method (the de-biasing step) is not used because the restriction of the normal equations to the identified terms is not well conditioned.
No de-biasing is used because the restriction of the matrix M k=1 FT w(tk)Fw(tk) onto the support set I is ill conditioned.

The method identifies the correct terms but does not approximate the correct values of the coefficients because the solution is far from homogenized.

The method identifies the correct terms and approximates the coefficients to within 0.005% in relative error. Even though the data are not completely homogenized and no de-biasing is used, the method can approximate the underlying uniform dynamics.

8.Travelling waves, solitons and ambiguity

As the coefficients and governing equations are learned from data, there is the potential for ambiguity when different equations generate similar datasets.

For example, consider the Korteweg–de Vries (KdV) equation:

$$u_t + u_{xxx} + 6(u^2)_x = 0,$$

whose solution is given by:

$$u(x, t) = 500\ \mathrm{sech}^2 \left(5 \sqrt{100}(x - 1000t - 1 /2 \right)$$

And the above equation is also a solution of the one-way wave equation

$$u_t + 1000u_x = 0$$

Model ambiguity is a potential issue with this approach when a large feature space is used.

As the one-way wave equation has fewer terms (a sparser governing equation), the method choose equation (2) over equation (1).

9. Additive noise on the data

- In above examples, velocity w^0_t was corrupted by additive Gaussian noise.
- In this example, Gaussian noise is added directly to the data themselves.

  Velocity is calculated from the noisy data.

Applying the method to Viscous Burgers' equation, produces accurate results before noise ≈3% and inaccurate results after ≈3% (the noise level is computed in the same way as before).

This is due to an inaccurate approximation to the temporal and spatial derivatives.

Data smoothing (or denoising):

- Needed to produce an accurate approximation.
- A smoothing approach is shown :

  -A smooth approximation u^s is constructed by u^s = argmin $\beta\|\nabla w\|^2 + \| w - u^n\|^2$ with a smoothing parameter $\beta > 0$. (u^n is noisy data)

  -The solution is given by $u^s = (I - \beta\Delta)^{-1}u^n$

  -The learning algorithm is then applied to the filtered data u^s .

This example shows a potential issue with this approach. As the data themselves are noisy, direct velocity estimation using finite differences will be unstable.

Conclusion

- This work presented L1 regularized least-squares minimization to select the correct features that learn the governing PDE.
- The identification of the terms in the PDE (form of model) and the approximation of the coefficients come directly from the data.
- Advantages:
  The numerical experiments show that the proposed method is robust to data noise and size, can capture the actual features of the data, and can perform additional tasks such as extrapolation, prediction, and (simple) homogenization.
- Disadvantages:
  This work shows a potential issue with this approach:
  1. When applied directly to noisy data.
  2. When different equations generate similar datasets (Model ambiguity).

Throwback into the previous paper

The paper Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations written by Maziar Raissi put forth a deep learning approach for discovering nonlinear partial differential equations from scattered and potentially noisy observations in space and time. In that paper We approximate both the solution u and the nonlinear function N with two deep neural networks. We define a deep hidden physics model f as:

$$f := ut - N(t,x,u,ux, uxx,\dots)$$

Derivatives of u with respect to time t and space x is obtained by using Automatic differentiation.
After that the parameters of neural networks u and N were learned by minimizing the sum of squared errors.

$${}^{M}\Sigma_{k=1}(|u(t^{i}, x^{i}) - u^{i}|^{2} + f(t^{i}, x^{i})|^{2})$$

Inference

In the current paper the solution of partial differential equations were found with the help of sparse optimization. The sparse optimisation helped to make out the coefficients in a more optimized way .
Along with that the use of LASSO and L1 norms were also discussed there.

In the previous paper nonlinear partial differential equations were solved with a deep learning approach using neural networks.
Now we can also reach into a solution when the model is unknown initially.
Now we need to find the solution of Differential equations using the inference obtained from the previous works.

THANK YOU