SAND2016-9483C

Using Tensor Theory to Embed Invariances: A Case Study from Turbulence Modeling

J. Ling
Sandia National Laboratories
Livermore, CA 94550
jling@sandia.gov

Abstract

For many physical systems, the governing equations are written in terms of tensor quantities. These equations often have known invariance properties. When designing a machine learning model for terms in these equations, it is desirable to embed these invariance properties into the model. This embedding can be achieved directly using concepts from representation theory. This process is demonstrated on the application case of turbulence modeling. A neural network architecture that embeds Galilean invariance is proposed and is shown to provide significantly improved results in comparison to a naive multi-layer perceptron that does not leverage the tensor invariance properties of this system.

1 Introduction

Tensors arise naturally in the governing equations of many physical systems. For example, velocity is a first order tensor, the velocity gradient is a second order tensor, and the Hessian of the velocity is a third order tensor. Therefore, when applying machine learning to physical systems, tensors are the natural language.

Another key characteristic of many governing equations is that they have known invariance properties. For example, in classical dynamics, the laws of motion are invariant to inertial changes of frame of reference. This invariance property is also known as *Galilean invariance*. In simple terms, Galilean invariance means that the physics of a system do not change when the observer changes reference frame. Emmy Noether [1] showed how these invariance properties are linked at a fundamental level to conservation properties, such as conservation of mass and momentum. These invariance properties are a part of the basic physics of the system, and it is therefore critically important that any machine learning model for the system obey them: it would greatly decrease the reliability and credibility of a machine learning model if the model predictions depended on the user's choice of coordinate axes.

Because the governing equations are written in terms of tensor quantities, it makes sense to discuss these invariance principles in terms of tensor invariance. Tensor invariance is a well-studied field linked to representation theory and group theory. It provides a method of systematically creating an *integrity basis*, which is a spanning basis of tensors that respects an invariance property. This paper discusses a method of using concepts from tensor invariance theory to develop an innovative neural network architecture that has Galilean invariance embedded in it. This specialized network architecture is presented in the context of a case study in turbulence modeling described in Ref. [2]. This framework is quite general, and can be extended to embed other tensor invariance propertiess.

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

2 Problem Description

In turbulence modeling, the governing equations are the Navier Stokes equations. Directly solving these equations by discretizing them across a mesh is prohibitively computationally expensive for most flows of interest; even simple flows have been known to take hundreds of millions of core hours to simulate [3]. Most often, an approximate version of these equations, known as the Reynolds Averaged Navier Stokes (RANS) equations, is solved instead. The RANS equations can be solved much more computationally efficiently, but they have unknown terms that must be modeled. There is one term, the Reynolds stress anisotropy tensor $\mathbf{A}$, that is usually modeled as a polynomial function of the symmetric and anti-symmetric components of the velocity gradient tensor, $\mathbf{S}$ and $\mathbf{R}$ respectively. The most widespread model, also known as the Linear Eddy Viscosity Model (LEVM) proposes that $\mathbf{A}$ should be modeled as directly proportional to $\mathbf{S}$. While the LEVM is widely used because of its simplicity, it is known to give poor predictions of $\mathbf{A}$ in many flows of engineering relevance. Improving predictions of $\mathbf{A}$ has been at the crux of turbulence modeling for the past 40 years. In this case study, a machine learning model is developed to predict $\mathbf{A}$ as a function of $\mathbf{S}$ and $\mathbf{R}$.

3 Tensor Invariance

It is known that the Reynolds stress anisotropy tensor $\mathbf{A}$ obeys Galilean invariance. This means that it is invariant to translations at fixed velocity, rotations, and reflections of the coordinate axes. $\mathbf{S}$ and $\mathbf{R}$ are both already insensitive to translations at a fixed velocity, since they are based on the velocity gradient. On the other hand, both $\mathbf{S}$ and $\mathbf{R}$ change when the coordinate frame is rotated or reflected. Similarly, $\mathbf{A}$ is not expected to remain unchanged when the coordinate frame is rotated–it is expected to rotate with the coordinate frame. Mathematically, if we apply a rotation matrix $\mathbf{Q}$ to our coordinate frame, then we expect that:

$$\mathbf{A}(\mathbf{Q}\mathbf{S}\mathbf{Q}^T, \mathbf{Q}\mathbf{R}\mathbf{Q}^T) = \mathbf{Q}\mathbf{A}(\mathbf{S}, \mathbf{R})\mathbf{Q}^\mathbf{T} \tag{1}$$

A tensor $\mathbf{A}$ that obeys Eq. 1 for all rotation and reflection matrices $\mathbf{Q}$ and all input tensors $\mathbf{S}$ and $\mathbf{R}$ is Galilean invariant [4]. The group of all reflection and rotation matrices is called the full orthogonal group. Therefore, we require that $\mathbf{A}$ be invariant to the full orthogonal group. By the Caley Hamilton theorem, given an arbitrary finite set of input tensors, it is possible to construct a finite basis of invariant tensors, called an *integrity basis* [5]. Any invariant tensor function of these input tensors will be a linear combination of the integrity basis tensors. These integrity bases have been tabulated for most of the common invariance groups [6, 7, 8, 9].

In the case of turbulence modeling, where we have two tensor inputs $\mathbf{S}$ and $\mathbf{R}$, the integrity basis has been previously derived by Pope [5]. It consists of 10 tensors $\mathbf{T}^{(n)}$ for $n = 1, ..., 10$. We can express $\mathbf{A}$ as:

$$\mathbf{A} = \sum_{n=1}^{10} f^{(n)}\mathbf{T}^{(n)} \tag{2}$$

In Eq. 2, the tensors $\mathbf{T}^{(n)}$ are known functions of $\mathbf{S}$ and $\mathbf{R}$. The coefficients $f^{(n)}$ are unknown functions of the 5 scalar invariants of $\mathbf{S}$ and $\mathbf{R}$. Therefore, the goal of the machine learning model is to determine the coefficients $f^{(n)}$.

4 Embedding Invariance into the Network Architecture

In order to embed Galilean invariance directly into the model predictions, a neural network architecture was developed based on Eq. 2. This network will be referred to as the Tensor Basis Neural Network (TBNN), and is shown schematically in Fig. 1.

There were two input layers: Input Layer α and Input Layer β. Input Layer α accepted 5 inputs: the 5 scalar invariants of $\mathbf{S}$ and $\mathbf{R}$ specified in Ref. [5]. Input Layer α then fed into a set of densely connected hidden layers. The final hidden layer, Hidden α, had 10 neurons and linear activation functions, and represented the ten scalar coefficients $f^{(n)}$ in Eq. 2. Input Layer β read in the 10 tensors $\mathbf{T}^{(n)}$, which can be directly calculated through known relations from $\mathbf{S}$ and $\mathbf{R}$, also specified in Ref. [5]. There was then a final multiplicative merge layer that read in the final hidden layer Hidden

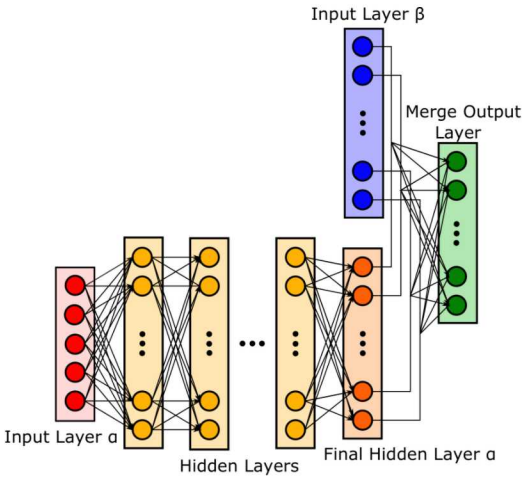

Figure 1: Schematic of TBNN

α and Input Layer β. This layer performed the multiplication and tensor addition specified in Eq. 2, yielding the prediction for $\mathbf{A}$.

Neural networks with multiplicative interactions, also known as *higher order neural networks*, have been previously investigated because of their biological analogs [10, 11] and because of their ability to embed structure and invariance properties into their mappings [12, 13]. In the multiplicative layer used in the TBNN, the output is $\mathbf{T}^{(1)}f^{(1)} + \mathbf{T}^{(2)}f^{(2)} + ... + \mathbf{T}^{(10)}f^{(10)}$, where $f^{(n)}$ were the inputs to the neuron from Hidden Layer α and $\mathbf{T}^{(n)}$ were the tensor inputs from Input Layer β.

In the hidden layers, a leaky Rectified Linear (ReLU) activation function was employed. Through Bayesian optimization [14], the following network hyper-parameters were determined: 8 hidden layers, 30 nodes per hidden layer, a learning rate of 2.5e-7.

The performance of the TBNN was compared to that of a generic Multi-Layer Perceptron (MLP) that had no embedded invariance properties. The MLP took as inputs the 9 distinct tensor components of $\mathbf{S}$ and $\mathbf{R}$. Through the same Bayesian optimization process, the MLP hyper-parameters were optimized to 10 hidden layers, 10 nodes per hidden layer, and a learning rate of 2.5e-6.

5 Data Sets

The TBNN and MLP were evaluated using a database of 9 flows for which both high fidelity as well as RANS simulation results were available. One of those flows was used as a validation set during the hyper-parameter tuning phase. The other 8 sets were used in leave-one-out cross-validation to evaluate the model performance. The data base included results from three different jet in crossflow configurations, flow around a square cylinder, flow around a wall-mounted cube, flow over a wavy wall, flow through a 2-D channel, flow through a square duct, and flow through a converging-diverging channel. More details on these 9 flow cases can be found in Ref. [2].

6 Results

The performance of the MLP and TBNN were evaluated through 8-fold cross-validation. For each cross-validation case, the Root Mean Squared Error (RMSE) was calculated on the validation set. The MLP had 1% higher error than the LEVM on average. Notably, the standard deviation of the relative change in RMSE was 26%, indicating that in some flows, the MLP provided much lower error than the LEVM, and in others, much higher error. Over the 8 cross-validation cases, the MLP provided reduced error in 4 of the cases and increased error in the other 4.

The TBNN out-performed both the MLP and the LEVM for all 8 cases, providing an average relative reduction in RMSE of 31% compared to the LEVM. The standard deviation of this reduction in

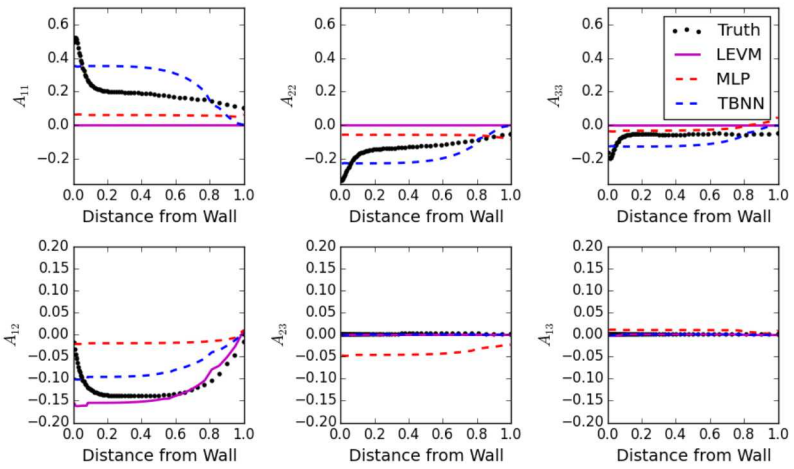

Figure 2: Predictions of the 6 distinct components of $\mathbf{A}$ for flow through a channel.

RMSE was 14%. These results demonstrate that the TBNN had overall better and more consistent performance than the MLP. Ling et al. [2] also showed that when the TBNN results for $\mathbf{A}$ were used in a flow solver, improved predictions of the mean velocity field were achieved.

Figure 2 shows the predictions of $\mathbf{A}$ for the flow through a 2-dimensional channel. In this flow, the only spatial variation is in the wall-normal direction. Since the tensor $\mathbf{A}$ is symmetric, there are only 6 distinct components in the tensor. As this figure shows, none of the models are able to accurately re-create the truth data for all 6 components. The LEVM does well on the off-diagonal components, but fails completely for the on-diagonal components of $\mathbf{A}$. The MLP does even worse, even failing to predict that the $\mathbf{A}_{23}$ and $\mathbf{A}_{13}$ components are uniformly zero because of the 2-dimensional nature of this flow. The TBNN predictions still differ significantly from the truth data, but overall give the closest predictions of the magnitude of the tensor components. These results show that the TBNN does vastly better than the MLP, but still has room for improvement.

7 Conclusions

Tensors are the natural format of inputs and outputs when applying machine learning to many scientific systems. The governing equations for dynamic systems have long been represented in terms of tensor expressions. These governing equations also have known invariance properties that the tensor terms must obey. Therefore, when using a machine learning model to predict one of these tensor terms, it is critical to enforce this invariance property. This paper presented a neural network architecture that allowed invariance properties to be embedded. This process was described in the context of a case study in turbulence modeling. The performance of a generic multi-layer perceptron with no embedded invariance was compared to that of a tensor basis neural network with embedded Galilean invariance. Through cross-validation, it was shown that the tensor basis neural network had consistently improved performance compared to the multi-layer perceptron. Furthermore, the tensor basis neural network significantly out-performed the conventional linear eddy viscosity model that is currently being used to simulate these flows. These very encouraging results demonstrate the utility of leveraging representation theory and tensor invariance theory to enforce known invariance properties.

Acknowledgements

We would like to acknowledge Kevin Matulef for his suggestions during the development of this paper. Supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND

References

[1] Noether, E., 1918. "Invariante variationsprobleme". *Nachrichten von der Gesellschaft der Wissenschaften zu Gottingen, mathematisch-physikalische Klasse*, pp. 235–257.

[2] Ling, J., Kurzawski, A., and Templeton, J., 2016. "Reynolds averaged turbulence modeling using deep neural networks with embedded invariance". *Journal of Fluid Mechanics*.

[3] Lee, M., and Moser, R., 2015. "Direct numerical simulation of turbulent channel flow up to $\mathrm{Re}_\tau \approx 5200$". *Journal of Fluid Mechanics*, pp. 395–415.

[4] Itskov, M., 2009. *Tensor Algebra and Tensor Analysis for Engineers*. Springer, Berlin.

[5] Pope, S., 1975. "A more general effective-viscosity hypothesis". *Journal of Fluid Mechanics,* ***72***, pp. 331–340.

[6] Spencer, A., and Rivlin, R., 1962. "Isotropic integrity bases for vectors and second-order tensors". *Archive for Rational Mechanics and Analysis,* ***9***, pp. 45–63.

[7] Smith, G., 1965. "On isotropic integrity bases". *Archive for rational mechanics and analysis,* ***18***, pp. 282–292.

[8] Spencer, A., 1987. "Isotropic polynomial invariants and tensor functions". In *Applications of tensor functions in solid mechanics*. Springer, Vienna.

[9] Zheng, Q., 1994. "Theory of representations for tensor functions–a unified invariant approach to constitutive equations". *Applied Mechanics Reviews,* ***47***, pp. 545–587.

[10] Schmitt, M., 2002. "On the complexity of computing and learning with multiplicative neural networks". *Neural Computation,* ***14***, pp. 241–301.

[11] Schnupp, J., and King, A., 2001. "Neural processing: the logic of multiplication in single neurons". *Current Biology,* ***11***, pp. 640–642.

[12] Giles, C., and Maxwell, T., 1987. "Learning, invariance, and generalization in high-order neural networks". *Applied optics,* ***26***, pp. 4972–4978.

[13] Reid, M., Spirkovska, L., and Ochoa, E., 1989. "Rapid training of higher-order neural networks for invariant pattern recognition". *IEEE International Joint Conference onNeural Networks*.

[14] Snoek, J., Larochelle, H., and Adams, R., 2012. "Practical Bayesian optimization of machine learning algorithms". *Advances in Neural Information Processing Systems*, pp. 2951–2959.