Manifold Vector Machine ($MVM^{©}$)

Hao

Summary

Translocating data to Manifold Vector Machine (MVM)is a *superpositioning* process. The essential architectural concern of MVM is to build a superpositioning scheme to achieve general improvements through each iteration process.

Iteration through datasets is a heavy duty for modern machines. This limit a wide range of machine learning methods, such as regression, interpolation, SVM, etc. If one exception, the fast developing neural network method, who overcome this issue with designed epoch, vectorisation, batching, etc., still face conflicting principles [0]. Reports have shown training on full ImageNet is possible today with models already achieved val-acc. 95% (val-acc.:validation set accuracy). But against expectations, these models performed *worse* in real use cases, often fall short 20% [1].

To summarise, the core concept of MVM is we can have a handful of choices to artificially define rules on how we iterate datasets. A related concept is Hinton's capsule [2] but looks at DIB-N (dimension increase by N). MVM should not be misunderstood as another DIB-N, because the dimension remain unchanged. It works with direct superposition, a concept of Quantum Computing. In the Use Cases session of this paper, you will see how MVM have achieved general improvements in performances even under strict measures.

Simple Rationale

The quality of data is at the heart of a model. When data is well prepared, there is a clear context. Iteration through each element of a *large* dataset follows monotonic marching that unavoidably result in diminishing clarity of context.

If it is agreeable a dataset *is* self-explainable of its context. Then ∃ a manifold to represent the definables using the existing dataset whether this process is of finite or infinite span. Then ∃ exist a logic chain that can be presented using a vector chain end-to-end(ETE). Suggest that this vector chain is with infinite length:

$$\vec{A}_0,\vec{A}_1,\vec{A}_2,\ldots,\vec{A}_n,\ldots,\vec{A}_\infty : \equiv \bigoplus_{n=0}^{\infty} \vec{A}_n \ (\|\vec{A}_n\| \neq 0, \forall n \in [0, *\infty] \ ) .$$

* $\ominus$: *Unified compact space, unit* = $\|\vec{A}_n\|$,

* *Infinity* ∞ *loosely defined at this moment,*
* *Use of comma as concatenation.*
* $\overrightarrow{A}_n$ *non-zero* $\|\overrightarrow{A}_n\| \neq 0$ *vector*

These vectors are defined with relative distance ETE, and their length $\| \bullet \|$, so that we have a sequence of *fixed ankle-joints* at each ETE point. As suggested the infinity is *loosely* defined. We are not sure the true behaviour of $\overrightarrow{A}_\infty$ except we know its a vector following the joint with $\overrightarrow{A}_{\infty-1}$, and the right-hand end point of the entire chain structure. A matrix representation of *vector* chain is shown below:

$$concat(\overset{\infty}{\underset{n=0}{\ominus}} \overrightarrow{A}_n) = (\overrightarrow{A}_0, \overrightarrow{A}_1, \overrightarrow{A}_2, \ldots, \overrightarrow{A}_n, \ldots, \overrightarrow{A}_\infty) = \begin{bmatrix} a_{00} & a_{01} & a_{02} & \cdots & a_{0n} & \cdots & a_{0\infty-1} & a_{0\infty} \\ a_{10} & a_{11} & a_{12} & \cdots & a_{1n} & \cdots & a_{1\infty-1} & a_{1\infty} \\ a_{20} & a_{21} & a_{22} & \cdots & a_{2n} & \cdots & a_{3\infty-1} & a_{4\infty} \end{bmatrix}$$

Since we have $\|\overrightarrow{A}_n\| \neq 0$ so that we cannot have *all-zeros* within the columns of this matrix. Therefore, the Column Rank is numeric equal to the total number of vectors, while Row Rank $\geqslant$ original dimension of the dataset. Suggest all vectors are already normalised with the same base, since they come from the same manifold of the original dataset. Then we have:

$$M_T \otimes concat(\overset{\infty}{\underset{n=0}{\ominus}} \overrightarrow{A}_n) = \lambda \times concat(\overset{\infty}{\underset{n=0}{\ominus}} \overrightarrow{A}_n).$$

* M_T : Transform Matrix
* λ: Transform Ratio

Here we introduced *discrete continuity* [3] since we wish to iterate the entire vector chain:

Definition:

> A *marching* is a mono-directional translocation activity from left-hand end of a vector joint to the right-hand end, then traveling through the entire vector chain in this fashion is one *oscillation denotable as* $\dot{O}$. Then we have:
>
> $$\text{Iteration } (d\delta) = \text{Floor}\left(\frac{N \times \dot{o}}{\|concat(\overset{\infty}{\underset{n=0}{\ominus}} \overrightarrow{A}_n)\|}\right)$$

* N: number of *oscillation through one marching.*
* Floor: guarantees each iteration is complete.

Therefore, we have iteration tangent space over transform matrix as:

$$\frac{dM_T}{d\delta} = \lambda \times M_T$$

This form is an iterable *chain* structure $\underset{iff}{\exists}$ a pair of ETE *fixed ankle-joints* for each $\overrightarrow{A}_n$. As you may see, λ is satisfied through inner product operator on M_T , and is iterable over the inner product space $\overrightarrow{\lambda} \cdot M_T$ where $\overrightarrow{\lambda} := (\lambda_0, \lambda_1, \ldots, \lambda_m)$ which is still definable with d_H (Hausdorff distance). Hence by definition:

Theorem:

$$\frac{d M_T}{d \delta} = \frac{\| \overrightarrow{\lambda} \cdot M_T \|}{d_H}$$

Where:

$$\exists\, d_{H total} := max\{min\{d(\overrightarrow{A}_0, \overrightarrow{A}_\infty)\,, d(\overrightarrow{A}_\infty, \overrightarrow{A}_0)\}\}\,,$$
$$\exists\, d_{H local} := max\{min\{d(\overrightarrow{A}_{\infty-1}, \overrightarrow{A}_\infty)\,, d(\overrightarrow{A}_\infty, \overrightarrow{A}_{\infty-1})\}\} = \|\overrightarrow{A}_n\|\,.$$

Special case ETE at $\overrightarrow{A}_0$ and $\overrightarrow{A}_\infty$:

$$\exists\, d_{H total} := \|\overrightarrow{A_0} - \overrightarrow{A_\infty}\|, \text{ iff } \bigoplus_{n=0}^{\infty} \overrightarrow{A}_n = \bigodot_{n=0}^{\infty} \overrightarrow{A}_n$$

As a result in each marching, there is a definable oscillation $\dot{o}$ with a reducible distance d_H. Interestingly, since our vector chain is in free form. We can artificially build manifold to hold the structure onto a durable coordinate system. This operation keeps the relativity of each vector and allows a definable conjunction to build a closed structure, the annotation of $\bigoplus_{k=0}^{\infty}$ is interchangeable with $\bigodot_{k=0}^{\infty}$ iff at the state i.e. two ends of the entire chain are met ETE, that satisfies d_H, this allows a reliable build for a continuous marching through a designed range of oscillations $\dot{o}$ *as* $\dot{O} = \sum_{0}^{\infty} \dot{o}$. In this state, we can build an iterator with maximised $\frac{d M_T}{d \delta}$ since $\max\{\|\overrightarrow{A_0} - \overrightarrow{A_\infty}\|, 0\} > 0$.

Before moving on to the algorithm section I would suggest building manifold for *single-entry* vector chains is rather computational expensive. The memory tools of modern computer systems for arrays and vectors are already very powerful. Building manifold architecture can be surprisingly fast and useful on large sets of data, rearrange multi-dimensional matrices. These structures are already computational heavy, so building manifolds with MVM will serve its purpose.

Examples:

Dot ($d_{H total} := \|\overrightarrow{0}\|$) , singularity definable with MVM. Although theoretically important as the ultimate superposition, there is no clear applicable use:

$$\bigoplus_{n=0}^{0} \overrightarrow{A}_n = \overrightarrow{A}_0 = \overrightarrow{0}$$

Ring, **Loop**, **Sphere**, **Horn-Torus**, ***Boy's Surface** ($d_{H total} := \|\overrightarrow{A_0} - \overrightarrow{A_\infty}\|$) are highly practical, some are more NAN(not-a-number) resilient than others depends on the dataset. In addition, *tighter* manifolds has higher impact on data point's local regions after a full march over one oscillation that directly result in vector beams aggregation, this is similar to the *amplitude amplification* process of quantum superpositioning:

$$\ominus_{n=0}^{\infty} \overrightarrow{A}_n \equiv \circledcirc_{n=0}^{\infty} \overrightarrow{A}_n \text{ , where locally } |\varphi_i\rangle = \sum \|\overrightarrow{A}_k\| \times |\circledast_k\rangle \text{ , for all} \overrightarrow{A}_{k\in(0,\infty)}$$

at the same *local region*, with each $\overrightarrow{A}_k$ as a *tendency* of a corresponding state $|\circledast_k\rangle$.

* An interesting hyper compactness (*hyper* superpositioning) resulted on Boy's Surface that blackout some entire vectors with NAN coordinates.

Line ($d_{H\,total} := \sum_{n=0}^{\infty} \|\overrightarrow{A}_n\|$ or $\|\overrightarrow{\infty}\|^*$ loosely defined), analogy for standard array in modern machines:

$$\ominus_{n=0}^{\infty} \overrightarrow{A}_n = (\sum_{n=0}^{\infty} \|\overrightarrow{A}_n\|)\overrightarrow{A}_{k\in(0,\infty)} \text{ , iff } \infty \text{ definable}$$

Recipe:

Step One : Data Preparation

There is no best practice, you can start with:

```
Define Problem
      ↓
Adopt a Scheme
```

Step Two : *M V M* Build (UML)

Define a class with a `Manifold_Method`

```
Manifold Method (Data,*args)
```

```
For data in Data:

  x = data.x
  y = data.y

  // MANIFOLD BUILD //

  /* coordinate map */
  xt,yt,zt = To_Sphere(x,y)
  /* dynamics */
  Twist(xt,yt,zt)
  Attractor(xt,yt,zt)
  /* hyper-tuning */
  xt*= args[0]
  yt*= args[1]
  zt*= args[2]
  /* output */
  Vector3 build = Vector3(xt,yt,zt)

  // VECTOR BEAM //

  /* init beam origin */
  Vector3 origin = Vector3.ones(3)
  /* polar coordination */
  Vector2 beam = Polarize(build,origin)

  // FETCH //

  data = beam
```

```
Class Manifold(Data)
```

```
__init__ Method ()
Manifold_Method:Method(Data,*args)
```

* As you can see, the *M V M* is easy to apply. There is no magic and the rationale can be applied with fine tuning techniques borrowed from the concepts in differential geometry. The ending part of this paper is going to present you with some use cases and a prototype application.

$DMA^{©}$ (Data Manifold Analysis) Use Cases:

Case One: $MVM^{©}$ as an Amplifier for Image Processing

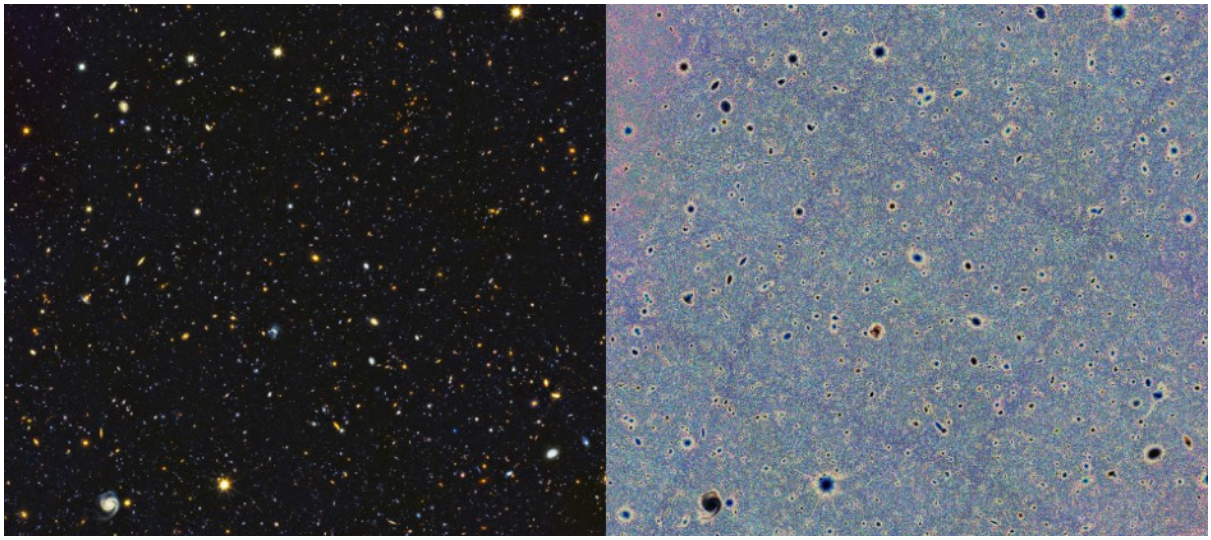

Figure: Hubble Deep UV (HDUV) Legacy Survey image (15k galaxies; August 16, 2018) MVM shows Illusive Grid Structure (All rights reserved © Hao)

Case Two: $MVM^{©}$ as a Tensor Layer for training CNN with ResNET50 / InceptionResNetV2 Bases

Setup:

The setup comprises a standard test set with 800 images, a validation set of 80 images with corresponding 40 labeled categories. I ran through three epochs through the architectures with parallel settings and comparable measures. All flattened, densed before applied either to standard activation with LeakyRELU activation or MVM (so there is no activation for MVM DMA architecture), the last layer is softmax with exactly the same settings. As for measurements, both accuracy and loss values are historically logged per epoch. As shown below, MVM is not only applicable as a standard layer, the performance is very neat, without adding any additional parameterisation, hardware upgrades, model complexity or increase of depth. The only issue as mentioned before is the result of NAN in more compact manifolds, vectors beams are been black-out as hyper superpositioning occurred, which however can be easily regularised through simple filtration.

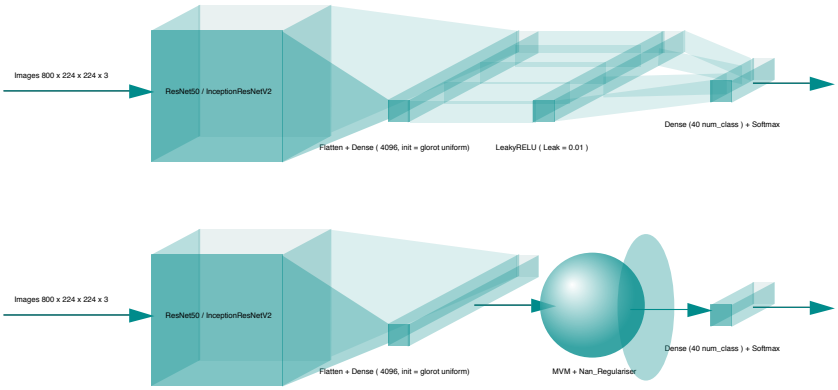

Figure : Architecture Comparison of the MVM^{-} (Above) and MVM^{+} (Below). (macSVG) (All rights reserved © Hao)

Result:

In order to compare the architectures in core functionality, subsequent optimisation are not been used, batching is full dataset, epoch is only set to three. GPU is not used, instead, both architecture were running on single core Intel Core i7 with Mac High Sierra OS. Historical *.csv log were used with native Tensorflow *CSV* logger together with full weighted network checkpoints for the epochs in *.h5. Four standard measures were taken, acc., loss, val_acc. For the test sets, and val_loss for the validation sets.

As for reading, in general, low loss ensures preservation of learning capacity for *true deep* learning, higher acc. in test set implies better performance through a designed iteration scheme, higher val-acc. is the essential measurement for model quality. * In addition, low time cost assures economic value , MVM^+ is in average 8 times faster than MVM^- on exactly parallel settings.

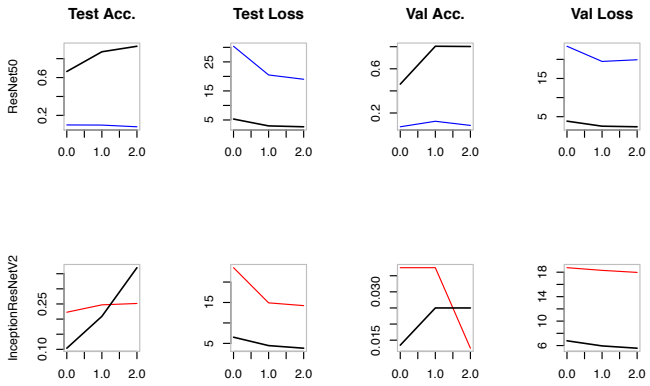

Figure: Performance Comparison of the MVM^- (Blue, Red) and MVM^+ (Black). (R native plot) (All rights reserved © Hao)

Discussion:

Artificial Intelligence is a fast growing interdisciplinary field of Science and Engineering. With MVM, I can build interesting architectures not-yet-fully tested, while at the sometime superior than its predecessors under standard performance measures. It is not by accident to construct virtual architecture for dataset, it is not by chance to journey through quantum phenomenons and differential geometry to resolve computational issues. With ever-progressing models, here we have an alternative start point with MVM, that will potentially allow us to build next-generation manifold AI not far from today.

Online Resources:

A NodeJs application that graphically present a few constant, linear, and non-linear functions using MVM algorithm to build a collection of 1e5 discrete unit points onto a non-orient torus manifold (https://dma-presentation-tool.herokuapp.com).

A MIT license Conversion Tool to simply transform your dataset to a MVM. (*Coming Soon*)

An algorithm patented Python package with code source available at request. (*Coming Soon*)

Reference:

[0] Zachary C. Lipton, Jacob Steinhardt, Troubling Trends in Machine Learning Scholarship, arXiv:1807.03341.

[1] Pierre Stock, Moustapha Cisse, *ConvNets and ImageNet Beyond Accuracy: Understanding Mistakes and Uncovering Biases*, arXiv:1711.11443.

[2] Geoffrey Hinton, *What's wrong with convolutional nets?*, techtv.mit.edu.

[3] Mark Burgin, *Continuity in Discrete Sets*, arXiv:1002.0036, 2010.